НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ "КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМ. ІГОРЯ СІКОРСЬКОГО"

"Інститут прикладного системного аналізу"
(повна назва інституту/факультету)

Системного проектування
(повна назва кафедри)

"На правах рукопису"
УДК 004.75

"До захисту допущено"

Завідувач кафедри _____
А.І. Петренко
(підпис) (ініціали, прізвище)

—_____20__ р.

Магістерська дисертація

на здобуття ступеня магістра

зі спеціальності 122 – Комп'ютерні науки та інформаційні технології
(Системне проектування сервісів)
(код і назва)

на тему: Комп'ютерні ігри як засіб навчання і тестування

Виконав: студент VI курсу, групи ДА-62м
(шифр групи)

Марков Дмитро Костянтинович _____
(прізвище, ім'я, по батькові) (підпис)

Науковий керівник проф., д.т.н., Петренко А.І. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант Розроблення стартап-проекту проф., д.т.н., Петренко А.І _____

Рецензент проф., д.т.н., Забара С. С. _____

Нормконтроль доцент, к.т.н., Кисельов Г. Д. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань. Студент

_____

Київ – 2018 року

Національний технічний університет України “Київський політехнічний інститут ім. Ігоря Сікорського”

Факультет (інститут) ННК ―Інститут прикладного системного аналізу
(повна назва)

Кафедра Системного проектування
(повна назва)

Рівень вищої освіти Другий (Магістерський)
(перший (бакалаврський), другий (магістерський) або спеціаліста)

Спеціальність 122 – Комп’ютерні науки та інформаційні технології (Системне проектування сервісів)
(код і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри _____
А.І. Петренко
(підпис) (ініціали, прізвище)

“___”_____2018 р.

ЗАВДАННЯ на дипломний проект (роботу) студенту
Маркову Дмитру Костянтиновичу
(прізвище, ім‘я, по батькові)

1. Тема проекту (роботи) “Комп’ютерні ігри як засіб навчання і тестування”

керівник проекту (роботи) Петренко А.І., д.т.н., проф.,
(прізвище, ім‘я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від “27“ березня 2018 р. №

2. Термін подання студентом дисертації . .2018

3. Об‘єкт дослідження: Комп’ютерні ігри

4. Предмет дослідження: Застосування ігор в навчальних цілях

5. Перелік завдань, які потрібно розробити
   1. Проаналізувати можливості використання комп’ютерних ігор в навчальних цілях
   2. Проаналізувати існуючі ігрові двигуни для розробки

3. Створити на вибраному ігровому двигуні ігри-прототипу з використанням програмування на C++
4. Створити демонстраційну презентацію розробленої гри
5. Створити курс розробки комп'ютерної гри для студентів старших курсів
6. Оформити роботу на основі отриманих результатів

6. Орієнтовний перелік публікацій

7. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Реалізація стартап-проекту			

8. Дата видачі завдання 01.02.2018

Календарний план

№ з/п	Назва етапів виконання дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Отримання завдання	01.02.2018	
2	Збір інформації та аналіз літератури	15.02.2018	
3	Вибір ігрового двигуна	28.02.2018	
4	Розробка гри	01.03.2018	
5	Розробка програми курсу створення гри	20.04.2018	
6	Оформлення дипломної роботи	05.05.2018	

Студент _____ Марков Д.К.
(підпис) (ініціали, прізвище)

Керівник проекту (роботи) _____ Петренко А.І.
(підпис) (ініціали, прізвище)

РЕФЕРАТ

НА МАГІСТЕРСЬКУ ДИСЕРТАЦІЮ

виконану на тему: Комп’ютерні ігри як засіб навчання і тестування

студентом: Марковим Дмитром Костянтиновичем

Робота виконана на 167 сторінках, містить 88 ілюстрацій, 22 таблиці. При підготовці використовувалась література з 31 джерела.

Актуальність теми

Прогрес не зупиняється ні на хвилину, і кожен день старі технології змінюються новими. Саме тому в сучасному світі важко переоцінити важливість навчання і особливо здібності навчатися. За останні декілька десятиліть технології змінилися докорінно, і це вже відобразилось на багатьох аспектах навчання в університеті. Проте, для того, щоб продовжувати створювати актуальні та висококваліфіковані кадри, покращуватись потрібно постійно.

З огляду на кількість провідних шкіл для розробки відеоігор, не дивно, що рішення про продовження навчання в цій дуже перспективній галузі виявилося настільки популярним серед студентів. Вражаюча кількість добре профінансованих і високоповажних шкіл розробки відеоігор у США надає дивовижний набір можливостей та кар'єрного потенціалу для іноземних студентів, які прагнуть зробити їхню любов до ігор своєю професійною діяльністю.

Мета та задачі дослідження

Метою даної роботи є дослідження можливостей використання ігор для навчання і розроблення навчальної програми створення ігор для студентів старших курсів.

Рішення поставлених завдань та досягнуті результати

У даній роботі було запропоновано навчальну програму створення ігор на C++ за допомогою ігрового двигуна Unreal Engine 4 для студентів старших курсів. Дана програма курсу дозволить навчати висококваліфіковані кадри для ігрових компаній по всьому світу, але навіть якщо програмування ігор не буду основною

спеціальністю студента після випуску, все одно буде здобуто навички в побудові логіки, роботі зі штучним інтелектом, нетворк програмуванні, програмуванні рендеру, а також використанні високоефективних оптимізацій коду, що дозволить студенту бути бажаним працівником в багатьох організаціях.

Об’єкт досліджень

Комп’ютерні ігри

Предмет досліджень

Методи використання комп’ютерних ігор в навчальних цілях

Методи досліджень

Для вирішення проблеми в даній роботі використовуються методи аналізу і синтезу, системного аналізу, порівняння, логічного узагальнення результатів.

Наукова новизна

Наукова новизна роботи полягає у створенні першого в своєму роді в Україні курсу розробки комп’ютерних ігор та впровадженні його в кращому технічному університеті країни.

Практичне значення одержаних результатів

Отримані результати дозволяють вже з наступного семестру розпочати роботу в підготовці студентів з напрямку “Розробка комп’ютерних ігор”. В кінці курсу кожен студент отримає в своє портфоліо працю, що буде гідним показником навичок при прийомі на роботу. Також будуть отримані навички в побудові логіки, роботі зі штучним інтелектом, нетворк програмуванні, програмуванні рендеру, а також використанні високоефективних оптимізацій коду на реальному робочому прикладі.

Публікації

Марков Д. Порівняльний аналіз алгоритмів оптимізації рендерингу в сучасних комп’ютерних іграх / Марков Дмитро. // Міжнародний науковий журнал “Інтернаука” 2018 №7 (47), 1 том. Київ 2018 – Р. 55-62.

Марков Д. Покращене оклюзивне відсічення в сучасних комп’ютерних іграх / Марков Дмитро. // Міжнародний науковий журнал “Інтернаука” 2018 №7 (47), 1 том. Київ 2018 – Р. 63-67.

Ключові слова

Комп’ютерні ігри, розробка, програмування, AI, network, render.

РЕФЕРАТ

НА МАГИСТЕРСКУЮ ДИССЕРТАЦИЮ

выполненную на тему:

Компьютерные игры как средство обучения и тестирования

студентом: Марковым Дмитрием Константиновичем

Работа выполнена на 167 страницах, содержит 88 иллюстрации, 22 таблицы. При подготовке использовалась литература из 31 источника.

Актуальность темы

Прогресс не останавливается ни на минуту и каждый день изменяются старые технологии и появляются новые. Именно поэтому в современном мире сложно переоценить важность обучения и особенно способности обучаться. За последние несколько десятилетий вид технологий в корне изменился и это уже отобразилось на многих аспектах обучения в университете, но, для того что бы продолжать создавать актуальные и высококвалифицированные кадры, улучшаться нужно постоянно.

Смотря на количество ведущих школ разработки видеоигр, не увидительно, что решение про продолжение обучения в этой очень перспективной сфере оказалось настолько популярным среди студентов. Поражающее количество хорошо профинансированных и уважаемых школ разработки видеогр в США предоставляет увидительный набор возможностей и карьерного потенциала для студентов, которые стремятся перевести их любовь к компьютерным играм в профессиональную деятельность.

Цель и задачи исследования

Целью данной работы есть исследование использования игр для обучения и разработка учебной программы создания игр для студентов старших курсов.

Решения поставленных задач и полученные результаты

В данной работе было предложено учебную программу для создания игр на C++ с помощью игрового движка Unreal Engine 4 для студентов старших курсов.

Данная программа курса позволит обучать высококвалифицированные кадры для игровых компаний по всему миру, но даже если программирование игр не будет основной специальностью студента после выпуска, всё равно им будет получено навыки в построении логики, работе с AI, нетворк программировании, программировании рендера, а также использовании высокоэффективных оптимизаций кода, что позволит студенту быть желанным работником во многих организациях.

Объект исследований

Компьютерные игры

Предмет исследований

Методы использования компьютерных игр в обучающих целях

Методы исследований

Для решения проблемы, в данной работе используются методы анализу и синтезу, системного анализу, сравнения, логического обобщения результатов.

Научная новизна

Научная новизна работы состоит в создании первого в своём роде в Украине курса разработки компьютерных игр и внедрении его в лучшем техническом университете страны.

Практическое значение полученных

Полученные результаты позволяют уже со следующего семестра начать работу по подготовке студентов по направлению “Разработка компьютерных игр”. В конце курса каждый студент получает в своё портфолио работу, которая будет достойным показателем навыков при приеме на работу. Также будут получены навыки в построении логики, работе с искусственным интеллектом, программировании рендера, а также использовании высокоэффективных оптимизаций кода на реальном рабочем примере.

Публикации

Марков Д. Сравнительный анализ алгоритмов оптимизации рендеринга в современных компьютерных играх / Марков Дмитрий. // Международный научный журнал “Интернаука” 2018 №7 (47), 1 том. Киев 2018 – P. 55-62.

Марков Д. Улучшенное оклюзивное отсечение в современных компьютерных играх / Марков Дмитрий. // Международный научный журнал “Интернаука” 2018 №7 (47), 1 том. Киев 2018 – P. 63-67.

Ключевые слова

Комьютерные игры, разработка, программирование, AI, network, render.

ABSTRACT

ON MASTER'S THESIS

on topic: Computer games as learning and testing tools

student: Dmytro K. Markov

Work carried out on 167 pages containing 88 figures, 22 tables. The paper was written with references to 31 different sources.

Topicality

Progress does not stop for a minute and old technologies are substituting the old ones. That is why in the modern world it is difficult to overestimate the importance of learning process and especially the human ability to study. Over the last few decades' technologies have changed dramatically. This has already reflected in many aspects of the university curriculum, however, in order to continue graduating highly skilled professionals, it is necessary to improve continuously.

Given the number of leading schools specializing in the development of video games, it is not surprising that the decision to develop in this very promising field of study was so popular among foreign students. The impressive amount of well-funded and highly respected video game development schools in the United States offers an amazing set of career opportunities for international students who want to make their commitment to the lives of games in their professional activities.

Purpose

The purpose of this work is to explore the possibilities of using games for learning as well as developing a curriculum for creating games for senior students.

Solution

In this paper a training program for senior students was offered that involves creating games using C ++ with Unreal Engine 4. This course will provide training of highly skilled professionals for gaming companies around the world. However, even if game programming is not going to be the main field of an alumnus, the essential skills in

terms of logic development, working with artificial intelligence, programming, rendering, and using highly effective code optimizations will make any student a desirable employee in many organizations.

The object of research

Computer games

The subject of research

Usage of computer games for educational purposes

Research methods

To solve the problems listed in this paper we used methods of analysis and synthesis, system analysis, comparison, logical generalization of the results.

Scientific novelty

The scientific novelty of the work is to create the first of the kind in Ukraine course on the development of computer games and its implementation in the best technical university of the country.

The practical value of research

The results allow to begin the students' training course in the field of computer games development starting from the next semester. At the end of the course every student will have a project in their portfolio displaying their skills. Also, the experience in constructing logic, working with artificial intelligence, network programming, rendering programming and using highly effective code optimizations on a real working example will be acquired.

Publications

Markov D. Comparative analysis of algorithms for optimizing rendering in modern computer games / Markov Dmytro. // International scientific magazine "Internet Science" 2018 №7 (47), 1 volume. Kiev 2018 - P. 55-62.

Markov D. Improved occlusion culling in modern computer games / Markov Dmytro. // International scientific magazine "Internet Science" 2018 №7 (47), 1 volume. Kiev 2018 - R. 63-67.

Keywords

Computer games, development, programming, AI, network, render.

ЗМІСТ

СПИСОК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ ТА ТЕРМІНІВ

AI – artificial intelligence

UE4 – Unreal Engine 4

Рендер – збірна назва технологій обробки та виводу зображення в комп’ютерних іграх

Нетворк – збірка назва технологій роботи с сітями

Ігровий двигун – платформа для розробки ігор

ВСТУП

Серед проявів сучасної інформаційної культури вагоме місце посідають комп'ютерні ігри. Не таємниця, що школярі годинами проводять час перед екранами моніторів, захоплюючи планети, знищуючи монстрів, створюючи цивілізації й т.п. Комп'ютеризація суспільства розвивається стрімко. Над ігровими комп'ютерними програмами працюють тисячі висококваліфікованих фахівців, намагаючись зробити кожну гру якомога привабливіше й цікавіше. При цьому використовуються різноманітні сюжети, у тому числі й на історичну тематику. І після яскравого екрану учні опиняються на уроці, на якому образи створюються лише за допомогою слів учителя, невеликих за форматом зображень, обмеженої кількості картин на історичну тему, рідше — за допомогою відеофрагментів. Образи, породжені на уроці, явно програють образам комп'ютерним. Тому резонно поставити запитання — чому б учителеві не взяти комп'ютерні ігри в союзники? "Вікторія", "Епоха імперій", "Хрестоносці", "Європа", "Цивілізація", "День Перемоги", "Світ танків", "Світ військових літаків", "Протистояння", "Рим", "Козаки", "100 років війни", "Історія імперій", "Дипломатія" — цей далеко не повний перелік ігор на історичну тематику.

А що, якщо подивитися на ігри з іншого боку – для розробки ігор потрібно знати дуже багато речей. Комп'ютерні ігри поділяються за жанрами: екшени, шутери, аркади, спортивні, квести, рольові (RPG/MMORPG), перегони, симулятори, логічні, пригоди, стратегії та ін. Кожен із цих жанрів має свої особливості, та потребує унікальних знань для їх створення.

Наприклад для стратегій, із специфічних речей, потрібно особливо звернути увагу на штучний інтелект, оскільки дії комп’ютерного супротивника мають велике значення для досвіду від гри.

Для шутерів, важливу роль, окрім штучного інтелекту, відіграють анімації та рендер, оскільки потрібно добре розуміти що компанії світового масштабу виставили дуже високу планку цих технологій для жанру.

Майже будь який із жанрів може містити у собі мультиплеерну модель. Це означає що в компанії, яка розробляла проект, є цілий відділ, котрий займається нетворк взаємодією.

Це все показує, що ігри являють собою дуже складні і високонагружені системи, які розробляються десятками або навіть сотнями професіоналів в галузях програмування, дизайну, анімацій.

В ході представленої роботи буде розібрано існуючі ігрові двигуни, вибрано один із них і на його основі створено навчальну програму курсу “Розробки комп’ютерних ігор”, що повинна допомогти зробити перші кроки та подолати бар’єр входу в індустрію комп’ютерних ігор.

1 Навчання розробці ігор в світі

1.1 Університети в США

Програми розвитку відеоігор у Сполучених Штатах є одними з найвидатніших у світі. Зважаючи на кількість поважних програм у США, іноземні студенти повинні встановити певні критерії, щоб мати можливість знайти найкращий для себе варіант.

Не секрет, що популярність відеоігор продовжує підніматися, а від випадкових до відданих гравців, відеоігри тепер є в кожному домі в усіх частинах США. У цьому випадку не дивно, що багато іноземних студентів вирішили перетворити свою пристрасть на ігри в кар'єру у розробці відеоігор. Все більш популярним стає рішення про вивчення розробки відеоігор в США серед іноземних студентів, як те, яке обіцяє привабливу кар'єру, створюючи захоплюючі нові відеоігри з тренуванням у найсучасніших програмах розвитку ігор. З багатьох причин студенти-іноземці вирішують вивчати розробку відеоігор у США, головними з яких є посилання як на процвітаючу індустрію відеоігор у Сполучених Штатах, так і вражаюче число всесвітньо відомих провідних шкіл для розробки відеоігор, розташованих у країні. В цьому розділі буде визначено деякі провідні школи для розробки відеоігор, а також надати інформацію про установи.

Найкращі школи для розробки відеоігор в США високо оцінюються на міжнародному рівні внаслідок ряду факторів, таких як якість та суворість навчального плану, кількість активних і успішних проектів, з якими працює департамент, швидкість працевлаштування випускників та вплив випускників на галузь. Навчання в одній з провідних шкіл для розробки відеоігор в США дозволить іноземним студентам отримати ступені, які будуть високо цінитися при прийомі на роботу, як у США, так і за кордоном.

Існує ряд інституцій, з дуже гарними відкликами в опитуваннях щодо найкращих шкіл для вивчення розробки відеоігор у США. Такі популярні та добре вивчені інститути включають деякі з найбільш відомих імен у цій галузі. Серед

таких видатних інститутів - Технологічний інститут штату Джорджия, Державний університет штату Північна Кароліна, Каліфорнійський університет Санта-Крус та Університет Дрекселя.

Програма Ворсетського політехнічного інституту у Вустері, штат Массачусетс, зосереджується менше на огляді виробництва та аспектах керівництва командою, а більше на технічному програмуванні та навичках дизайну.

Окрім того, в Прінстонському огляді щорічно виділяються університети та їх відповідні відомства. Ці рейтинги створені, щоб дати студентам приблизний посібник, і студенти повинні також розглянути інші фактори при виборі школи, такі як доступні гранти на навчання, коефіцієнт студента до вчителя та рейтинг робочих місць випускників. Проте, щоб студенти мали як точку відліку, ось найвищі рейтинги провідних шкіл для розробки відеоігор у Сполучених Штатах за версією Princeton Review:

Кращі бакалавратні школи для розробки відеоігор в США

- Університет Південної Каліфорнії, Лос-Анджелес, CA [1]
- Університет штату Юта, Солт-Лейк-Сіті, штат Юта [2]
- Технологічний інститут DigiPen, Редмонд, штат Вашингтон [3]
- Мічиганський державний університет, Східний Лансінг, штат Мічіган [4]
- Вустерський політехнічний інститут, Вустер, штат Массачусетс [5]
- Університет Дрекселя, Філадельфія, штат Пенсільванія [6]
- Champlain College, Берлінгтон, VT [7]
- Рочестерський технологічний інститут, Рочестер, штат Нью-Йорк [8]
- Бекер коледж, Вустер, штат Массачусетс [9]

Кращі випускні школи для розробки відеоігор у Сполучених Штатах

- Університет Південної Каліфорнії, Лос-Анджелес, CA [10]
- Рочестерський технологічний інститут, Рочестер, штат Нью-Йорк [11]
- Массачусетський технологічний інститут, Кембридж, штат Массачусетс [12]
- Університет Центральної Флориди, Орландо, штат Флорида [13]
- Південний методистський університет, План, штат Техас [14]
- Університет Карнегі-Меллона, Пітсбург, штат Пенсільванія [15]
- Саванна коледж мистецтв і дизайну, Саванна, штат Джорджія [16]
- Технологічний інститут DigiPen, Редмонд, штат Вашингтон [17]

- Університет Каліфорнії, Санта-Круз, Санта-Круз, штат Каліфорнія [18]
- Університет Дрекселя, Філадельфія, штат Пенсільванія [19]

З огляду на кількість провідних шкіл для розробки відеоігор, не дивно, що рішення про продовження цієї дуже перспективної галузі навчання виявилося настільки популярним серед іноземних студентів. Вражаюча кількість добре профінансованих і високо поважних шкіл для розробки відеоігор у США надає дивовижний набір можливостей та кар'єрного потенціалу для іноземних студентів, які прагнуть зробити їхню прихильність до життя ігор у своїй професійній діяльності. Ті іноземні студенти, які вирішили вивчити розробку відеоігор у США, повинні вивчити деякі програми розробки відеоігор, що пропонуються в цих установах, щоб ознайомитись із конкретним підходом та зосередженням будь-якої майбутньої школи. [20]

1.2 Університети в Англії

1.2.1 Дизайн та програмування комп'ютерних ігор, Стаффордширський університет

Заснована в Сток-он-Трент, цей курс проводиться за підтримки TIGA, однієї з головних гральних влад Великобританії. Вона також охоплює більше однієї дисципліни у цій галузі (що ви побачите в інших університетах у цьому списку) - дизайн та програмування - так що коли ви закінчите курс ви будете свідомі в багатьох сферах розробки відеоігор. Це зручно для заснованих студій, які прагнуть наймати ширококваліфікованих співробітників та для випускників, які хочуть створювати власні студії з нуля. [21]

Крім того, студенти мають доступ до студії Rare's Motion Tracing, а студія ігрового дизайну в Стаффордширі фінансується Epic Games, що не потребує додаткової інформації. Також проводить регулярні події, відвідані фахівцями галузевих ігор; вони ідеально підходять для підбору покажчиків (і, можливо, вони виявляються під час навчання).

Цікаво, що Стаффордшир нещодавно представив VR-сфокусований дизайн курсу - з віртуальною реальністю ігри можуть перейти на якісно інший рівень. [22]

1.2.2 Розробка та розвиток ігор, Манчестерський університет метрології

Цей курс, що базується в самому серці Манчестера, пропонує типові дизайн-дослідження, а також більш широкий фокус на елементи, включаючи мобільні та соціальні ігри. Хочете зробити чергову Candy Crush? Ну, встаньте в чергу, але це тип курсу для вас. [23]

Манчестер Мет також пропонує варіант, що дозволяє відкласти навчання на третьому курсі під час перенесення на місце після закінчення курсу. Зрештою, досвід є неоціненним, і тисяча років навчання є нічим в порівнянні з 12 місяцями в робочій студії. [24]

Однак, що робить цей курс доречним для цього переліку, це місце розташування: Манчестер - це великий центр творчих індустрій, а також прекрасне місто в своєму розпорядженні. Окрім твердого обґрунтування дизайну ігор, ви насолоджуєтеся всіма перевагами прийняття Манса, це більше, ніж просто незвичний спокусник і схильність до Смітів. [25]

1.2.3 Дизайн ігор, Університет Брунеля

Під керівництвом професіоналів, курс Брунеля пов'язує теорію та практику дизайну ігор. Обсяг широкий, охоплює теорію дизайну, плюс більше практичних модулів і навіть мистецтво та бізнес-речі. Це програма, спрямована на те, щоб дати студентам гарну основу для основних елементів дизайну ігор.

Курс не спрямований на програмування / фахівців з інформатики, які не вимагають попереднього досвіду в цих дисциплінах. З цієї причини ви можете уникнути цього, якщо ви вже отримали підстави для розвитку - але це ідеально підходить для тих, хто хоче десь почати.

Курс Брюнеля, як і багато інших, вимагає представлення закінченого прототипу, тобто студенти повинні насправді робити щось із навичками, які вони набувають. [26]

1.2.4 Ігровий дизайн та мистецтво, Університет Саутгемптона

Курс Саусхемптона, заснований у Школі мистецтв Вінчестера, охоплює ті самі загальні теми інших курсів. Однак є одна велика різниця, що відокремлює її від інших курсів у цьому списку - "мистецтво" елемент ступеня назви. Це означає, що Саутгемптон пропонує кросовер з художньою графікою, образотворчим мистецтвом і курсами модного та текстильного дизайну в художній школі. Цей унікальний пункт продажу може зацікавитись тим, хто більше схиляється до мистецтва. У будь-якому випадку, міждисциплінарні курси - хоч і менш цілеспрямовані - пропонують

чудові шанси навчитися додатковим навичкам, які можуть зробити вас видатними для майбутніх роботодавців.

Крім того, ви знаходитесь поруч з дуже хорошим пляжем - якщо ви готові взяти півгодинний поїзд до Борнмуту, то є. [27]

1.2.5 Обчислення (ігри, бачення та взаємодія), Імперський коледж Лондона

Для тих, хто шукає курс, який спеціально не орієнтований на дизайн ігор, це такий. Це не означає, що Імперський курс - це вологий сквиб. Фактично, приймаючи це, ви отримаєте всесвітньо відомої університетської освіти, наповнене важким академічним вивченням найважливіших технічних факторів обчислювальної техніки, IT та всього цього прекрасного, складного речей.

Як чотирирічний курс, це довше, ніж інші в цьому списку (хоча в деяких випадках доступні варіанти бутерброда та неповного робочого дня) - додатковий рік використовується для навчання в області логіки, математики, компіляторів, мереж , візуалізація та багато інших елементів. Студенти вивчають багато речей, в основному.

Цей курс може не надавати доступ до лабораторій захоплення руху або вимагати створення гри для вашого остаточного проекту, але його класи дуже поважають і навчать вас набагато більше, ніж потрібно почати кар'єру в світі ігор. Проте, є недоліки, особливо, наскільки важко входити, і вартість проживання в Лондоні. [28]

1.3 Висновок

В даному розділі було розглянуто різні вищі навчальні заклади в яких є курси розробки відео-ігор. В ході дослідження було виявлено, що найбільш розвинена сфера розробки ігор в США та Великобританії. Було оброблено декілька різних

рейтингів університетів та вибрано кращі серед них. Деякі з представлених університетів були детально розписані та порівняні із своїми прямими конкурентами.

Загалом можна побачити, що провідні країни світу постійно розвивають розробку ігор, так як розуміють які перспективи лежать у цій галузі.

В США розробку ігор підтримують такі відомі заклади, як Массачусетський технологічний інститут, Університет Південної Каліфорнії, Університет штату Юта, Мічиганський державний університет, Рочестерський технологічний інститут, Технологічний інститут DigiPen. Це означає, що на навчання розробці комп'ютерних і мобільних ігор США витрачає десятки й сотні мільйонів доларів щорічно.

У Великобританії розробкою займаються такі відомі інститути, як Імперський коледж Лондона, Університет Саутгемптона та Стаффордширський університет.

2 Вибір двигуну для розробки гри

2.1 Існуючі ігрові двигуни та їх порівняльний аналіз

2.1.1 Ігровий двигун Unity

Unity — багатоплатформовий інструмент для розробки дво- та тривимірних додатків та ігор, що працює на операційних системах Windows і OS X. Створені за допомогою Unity застосування працюють під системами Windows, OS X, Android, Apple iOS, Linux, а також на гральних консолях Wii, PlayStation 3 і XBox 360.

Є можливість створювати інтернет-додатки за допомогою спеціального під'єднуваного модуля для браузера Unity, а також за допомогою експериментальної реалізації в межах модуля Adobe Flash Player. Застосування, створені за допомогою Unity, підтримують DirectX та OpenGL.

Технічні характеристики

Сценарії на C#, JavaScript та Boo;

Ігровий рушій повністю пов'язаний із середовищем розробки. Це дозволяє випробовувати гру прямо в редакторі;

- Робота з ресурсами можлива через звичайний Drag&Drop.
- Система успадкування об'єктів;
- Підтримка імпортування великої кількості форматів файлів;
- Вбудований генератор ландшафтів;
- Вбудована підтримка мережі;

Існує рішення для спільної розробки — Asset Server. Також можна використовувати зручний для користувача спосіб контролю версій. Наприклад, SVN або Source Gear;

Редактор Unity має простий Drag & Drop інтерфейс, який легко налаштовувати, що складається з різних вікон, завдяки чому можна проводити налагодження гри прямо в редакторі. Рушій підтримує три сценарних мови: C #, JavaScript (модифікація). Проект в Unity ділиться на сцени (рівні) — окремі файли, що містять свої ігрові світи зі своїм набором об'єктів, сценаріїв, і налаштувань. Сцени можуть містити в собі як, об'єкти (моделі), так і порожні ігрові об'єкти — тобто ті, які не мають моделі. Об'єкти, в свою чергу містять набори компонентів, з якими і взаємодіють скрипти. Також у них є назва (в Unity допускається наявність двох і більше об'єктів з однаковими назвами), може бути тег (мітка) і шар, на якому він повинен відображатися. Так, у будь-якого предмета на сцені обов'язково присутній компонент Transform — він зберігає в собі координати місця розташування, повороту і розмірів по всіх трьох осях. У об'єктів з видимою геометрією також за замовчуванням присутній компонент Mesh Renderer, що робить модель видимою.

Також Unity підтримує фізику твердих тіл і тканини, фізику типу Ragdoll (ганчіркова лялька). У редакторі є система успадкування об'єктів; дочірні об'єкти будуть повторювати всі зміни позиції, повороту і масштабу батьківського об'єкта. Скрипти в редакторі прикріплюються до об'єктів у вигляді окремих компонентів.

При імпорті текстури в рушій можна згенерувати alpha-канал, mip-рівні, normal-map, light-map, карту відображень, проте безпосередньо на модель текстуру прикріпити не можна — буде створено матеріал, з яким буде призначений шейдер, і потім матеріал прикріпиться до моделі. Редактор Unity підтримує написання і редагування шейдерів. Крім того він містить компонент для створення анімації, яку також можна створити попередньо в 3D-редакторі та імпортувати разом з моделлю, а потім розбити на файли. [29]

2.1.2 Ігровий двигун Unreal Engine 4

Unreal Engine — ігровий рушій, розроблюваний і підтримуваний компанією Epic Games.

Перша гра, створена на цьому рушію — Unreal — з'явилася 1998року. З тих пір різні версії цього ігрового рушія використали в більш ніж сотні ігор, серед яких Deus Ex, Lineage II, Thief: Deadly Shadows, Postal 2, серіях ігор Brothers in Arms, серія ігор Splinter Cell, Tom Clancy's Rainbow Six, а також у відомих ігрових серіях Unreal і Unreal Tournament від самих Epic Games. Пристосований у першу чергу для шутерів від першої особи, рушій використовувався й при створенні ігор інших жанрів.

Написаний мовою C++, рушій дозволяє створювати ігри для більшості операційних систем і платформ: Microsoft Windows, Linux, Mac OS і Mac OS X, консолей Xbox, Xbox 360, PlayStation 2, PlayStation Portable, PlayStation 3, Wii, Dreamcast і Nintendo GameCube. У грудні Марк Рейн продемонстрував роботу рушія Unreal Engine 3 на iPod Touch і iPhone 3GS. У березні 2010 робота рушія була продемонстрована на комунікаторі Palm Pre, що базується на мобільній платформі webOS.

Для спрощення портування рушій використовує модульну систему залежних компонентів: підтримує різні системи рендерингу (Direct3D, OpenGL, Pixomatic; раніше підтримувалися Glide API, S3 Metal, PowerVR SGL), відтворення звуку (EAX, OpenAL, DirectSound3D; раніше підтримувалися A3D), засоби голосового відтворення тексту, розпізнавання мовлення (тільки для Xbox360, PlayStation 3, Nintendo Wii і Microsoft Windows, також планувалося для Linux і Mac), модулі для роботи з мережею й підтримка різних пристроїв вводу.

Для гри у мережі підтримуються технології Windows Live, Xbox Live, і GameSpy, включаючи до 64 гравців (клієнтів) одночасно. Попри те, що офіційно засоби розробки не містять у собі підтримки великої кількості клієнтів на одному сервері, рушій використовувався для створення MMORPG-ігор. Один з найвідоміших представників жанру, Lineage II, використовує рушій Unreal Engine.

Усі елементи ігрового рушія представлені у вигляді об'єктів, що мають набір характеристик, і клас, який визначає доступні характеристики. У свою чергу будь-

який клас є “дочірнім” класом **object**. Серед основних класів і об'єктів можна виділити наступні:

Актор (**actor**) — базовий клас, що містить усі об'єкти, які мають відношення до ігрового процесу й мають просторові координати.

Павн, пішак (**pawn**) — фізична модель гравця або об'єкта, керованого штучним інтелектом. Назва походить від англ. *pawn* — той, ким *маніпулюють* (*pawn* можна перевести також як *пішак*, тому такий об'єкт без якої-небудь моделі виглядає як пішак). Метод керування описаний спеціальним об'єктом, такий об'єкт називається *контролером*. Контролер штучного інтелекту описує лише загальну поведінку пішака під час ігрового процесу, а такі параметри як “здоров'я” (кількість пошкоджень, після яких пішак перестає функціонувати) або, наприклад, відстань, на якій пішак звертає увагу на звуки, задаються для кожного об'єкта окремо.

Світ, рівень (**world**, **game level**) — об'єкт, що характеризує загальні властивості “простору”, наприклад, силу тяжіння й туман, у якому розташовуються всі актори. Також може містити в собі параметри ігрового процесу, як, наприклад, ігровий режим, для якого призначений рівень.

Для роботи із простими й, як правило, нерухомими елементами ігрового простору (наприклад, стіни) використовується бінарна розбивка простору — увесь простір ділиться на “заповнене” і “порожнє”. В “порожній” частині простору розташовуються всі об'єкти а також тільки в ній може перебувати “точка спостереження” при відмальовці сцени. Можливість повного або часткового поміщення об'єктів в “заповнену” частину простору не виключається, однак може привести до неправильної обробки таких об'єктів (наприклад, розрахунки фізичної взаємодії) або неправильної відмальовки у випадку поміщення туди “точки спостереження” (наприклад, ефект “залу дзеркал”). Усі пішаки, що потрапляють в “заповнену” частину простору, відразу “гинуть”.

Поверхня (**surface**) є основним елементом бінарного дерева простору. Ці елементи створюються на грані перетинання між “заповненою” і “порожньою”

частинами простору. Група елементів бінарного дерева простору називається нодом (**node**, укр. *вузол*). Цей термін, як правило, уживається в контексті **node count** — кількість нодів на екрані або в ігровому просторі взагалі. Кількість нодів, одночасно видимих на екрані впливає на продуктивність при промальовуванні сцени. Якщо якийсь нод не потрапляє на екран або перекривається цілком іншими нодами, він не обраховуються — це служить для підвищення продуктивності, особливо в закритих просторах. Розбивка всього простору на групи нодів називається зонуванням. Для цього іноді використовуються *портали* — невидимі поверхні, які служать для того щоб вручну розділити великий нод на два менші. Крім порталів використовуються *антипортали*, які обмежують області відмальовки.

Опис “заповнених” і “порожніх” частин простору виконується за допомогою набору замкнених тривимірних об'єктів, складених з не пересічних поверхонь — брашей (**brush**, укр.*пе́нзель*). Цей принцип побудови простору називається конструктивною суцільною геометрією. Геометрія може бути “аддитивною” (увесь простір початково “порожній”) і “віднімальною” (початково заповнений матерією простір). Браши діляться на три типи:

- Суцільні (**solid**) — повноцінно беруть участь у бінарній розбивці простору.
- Аддитивні (**additive**) — “заповнюють” бінарний простір.
- Віднімальні (**substractive**) — “вирізують” об'єми у просторі.
- Напів-Суцільні (**semi-solid**) — не впливають напряму на бінарне дерево простору, однак впливають на її фізичну модель. Можуть тільки “заповнювати” простір. Слугують для створення “невидимих” перешкод, а також зниження числа полігонів і нодів.
- Порожні (**non-solid**) — тільки створюють поверхні, не впливають на бінарне дерево простору. Використовуються переважно для створення *об'ємів* (**volume**) — частина простору, яка має властивості, відмінні від властивостей ігрового світу. Об'єми мають пріоритет, властивості об'єму з великим пріоритетом застосовуються до акторів, що

перебувають у ньому. Ігровий світ завжди має мінімальний пріоритет. За допомогою об'ємів можна змінити гравітацію, в'язкість, туман і таке інше. Об'єми, починаючи з версії рушія Unreal Engine 2, використовуються для створення води (але не водної поверхні). [30]

2.1.3 Ігровий двигун Id Tech 4

ID Tech 4, відомий як рушій гри для Doom 3, розроблений Id Software і вперше використаний в відеогрі Doom 3. Рушій був розроблений Джоном Кармаком, який також створив попередні системи, такі як ті, Doom і Quake, які також широко відомі через значні успіхи в цій області. Цей OpenGL був надав рушію також можливості використовуватися в Quake 4, Prey, Enemy Territory: Quake Wars, Wolfenstein і Brink. Рушій був випущений як повністю комерційний продукт, доступний для ліцензування стороннім компаніям, проте після виходу id Tech 5, **id Tech 4** був виданий як вільне ПО. Рушій "**id Tech 4**" використовує OpenGL як інтерфейсу програмування додатків (англ. *API*). **ID Tech 4** має всеосяжну мову сценаріїв, яка може бути використана при створенні модів.

У порівнянні з попереднім і широко використовуваним id Tech 3 (движок Quake III Arena) та id Tech 2 (двигун Quake II), id Tech 4 мав менший успіх у ліцензуванні третіх сторін. Це особливо очевидно у порівнянні з найближчими сучасниками Unreal Engine 2 (2002) та Unreal Engine 3 (2006) від Epic Games. id Software не бажає ліцензувати свій найновіший движок, перш ніж його "батьківська гра" Doom 3 була завершена. Проте неочікуваний тривалий час розробки в Doom 3 з 2002-04 років означав, що вони не змогли зіткнутися з Unreal Engine 2 Epic Games у цей період. Таким чином, багато хто, хто отримав ліцензію на Unreal Engine 2, змогли легше перейти на Unreal Engine 3.

Незважаючи на те, що id Tech 4 прийняв нове напрямок із динамічним світлодіодним освітленням, ця нестандартна функція мала більш жорсткі вимоги до апаратних засобів і була спочатку корисною лише в "приголомшливих іграх" (до моменту додавання MegaTexture), тоді як дедалі більша кількість розробників

віддавали перевагу традиційним двигунам що може призвести до великих відкритих площадок. Також помітним був відносний відсутність технології iD Tech 4 в порівнянні з конкуруючими двигунами FPS; id Tech 4 зазвичай вимагає сумісного з DirectX 8.0 графічного процесора, такого як GeForce 3; конкуруючий джерело двигуна (який був розроблений з попереднього двигуна GoldSrc) все ще може працювати на старих поширених GPU DirectX 7 (хоча і без використання шейдерів). [31]

2.2 Вибір ігрового двигуна для створення гри

3 ключові пункти для кожного двигуна.

- Usability (користувальницький інтерфейс, як легко було вчитися і розвиватися)
- Функціональність (що саме двигун може зробити)
- Ціна (говорить сам за себе)

2.2.1 Ігровий двигун Unity

Це голіаф незалежної розробки ігор, і останнім часом великі студії почали використовувати його і з вагомими причинами. Я вперше використовував Unity в 2012 році, але це було фактично випущено на Apple Worldwide Developers Conference ще в 2005 році. Я почав використовувати його в той час, коли вийшла версія 4.0.

До того моменту, коли я почав використовувати Unity, я вже використовував такі інструменти, як Game Maker та Game Salad, але все ще був відносно новим для розробки ігор і продовжував вивчати мотузки. Єдність, власне, тільки почала справді тягтися за допомогою встановлених студій розробки ігор. Будь-яка студія, з якою я розмовляв, використовувала Unity - і це правильно. Unity має фантастичний інтерфейс, який дозволяє розробнику ефективно керувати проектом з програми get-go. Він безкоштовний, і ви навіть можете публікувати свою гру на певних платформах без ліцензії. Це приватно для кожної молодої студії. Ті, хто тільки починає роботу і не можуть дозволити собі дорогі ліцензії.

Це дуже добре закруглений двигун. Більшість двигунів обслуговують 3-х або 2-дводні розробки - не так багато підтримують обидва. Так, ви можете робити 2-х ігор з 3-дюймовим двигуном, але 9 разів з 10 ви краще з двома конкретними двигунами. Єдність змінює це - це дозволяє будувати обидва.

Хоча "Єдність" відносно легко підібрати та використовувати, вона має криву навчання для новачків. Тим не менш, у нього є велика кількість онлайн-підручників, чудової документації та спільноти людей, які хочуть допомогти. Це полегшує друк на декількох платформах за допомогою одного кліка, а підтримувані платформи включають ПК, мобільні пристрої, великі консолі типу PS4, Xbox One і Nintendo Wii U. Він також підтримує кілька мов, таких як C #, JavaScript та Action Script 3.

В цілому це надзвичайно надійно і дуже рекомендується. Рейтинг 8/10.

2.2.2 Ігровий двигун Unreal Engine 4

Unreal Engine 4 - це Epic ігри, власний двигун, розроблений в будинку. Перша версія була випущена в 1998 році. Цей движок використовувався для розробки Unreal Tournament. Epic випустив четверту ітерацію двигуна Unreal для суспільного використання в березні 2014 року. Я не дуже використовував цей двигун. У мене трохи більше досвід роботи з UDK 3. Що я можу сказати, це те, що Unreal Engine використовується для виробництва деяких найвищих у світі ігор AAA на ринку.

Ігри, схожі на серію Batman Arkham на останніх консолях, "Epics", серії "Gears of War", були зроблені на udk3, а також інді-хіти, такі як Toxic Games "Qube". Зовсім недавно на Unreal Engine 4 були розроблені ігри типу "Денне світло" для ПК та PS4 від Zombie Studios та інше Epic Games назва "Fortnite".

Якість цих титулів говорить про себе. Мій особистий досвід роботи з Unreal полягав у тому, що він сильно розбився. Мені довелося врятувати свій проект після кожної невеликої зміни, щоб уникнути цього. Це головним чином тому, що моєму ПК не вдалося запустити движок. Це єдиний двигун, який я використовував, викликав цю проблему. В цілому, мені дуже сподобалося, і хотілося б потрапити у трохи більше, але постійне збої було величезним відключенням. Однак користувальницький інтерфейс і елементи керування навігацією є дуже полірованими та прості у використанні.

Ціновий пункт був безкоштовним, доки гра не нараховувала певну суму грошей, тоді розробник повинен був оплатити ліцензію. Нова модель - це підписка, починаючи від 19 доларів на місяць. Існує також велика крива навчання, тому не підходить для нових починаючих розробників ігор.

Загалом це очевидно, що це найпотужніший двигун для великих студій, але, можливо, не ідеально підходить для невеликих команд Indie з низьким бюджетом. Рейтинг: 9/10

2.2.3 Ігровий двигун Id Tech 4

Початкова вимога id Tech 4 полягала в тому, що для цього потрібен високопродуктивний графічний процесор (GPU) з повністю програмованими вершинами та піксельними шейдерами, такими як Nvidia GeForce 3 або ATI Radeon 8500, що має принаймні 64 МБ VRAM. За E3 2002 рекомендований GPU був "100% сумісний з DirectX 9.0b", такими як Radeon 9700 з 128 Мб VRAM. Хоча функції Radeon 9700 DirectX 9.0 не потрібні для відтворення гри, його просунута архітектура, 256-розрядна шина пам'яті та ефективність необхідні для запуску Doom 3 з високою деталізацією та грати зі швидкістю. Графічний режим "Ультра", що входить до складу Doom 3, навіть не буде працювати на нинішніх популярних відеокартах, доступних у 2004 році, що вимагає щонайменше 512 МБ відеопам'яті для правильного відображення та відтворення за швидкістю, що означає, що воно було в основному непридатним під час роботи випуск

id Tech 4 призвело до виснаження графічних чіпів DirectX 7, таких як широко поширені GeForce 2 і Radeon 7200, а також старі набори чіпсетів, такі як RIVA TNT2 і Rage 128, і рендеринг програмного забезпечення (з інтегрованим Intel GMA). До появи технології id Tech 4 потужний процесор зміг дещо компенсувати стару відеокарту. Хоча Джон Кармак спочатку застеріг геймерів не купувати GeForce 4 MX (які випадкові споживачі часто плутають з DirectX 8 здатні GeForce 4 Ti, хоча це

був у кращому випадку поліпшений GeForce 2), його дещо широке застосування, змушене ID Software, щоб додати його список підтримуваних карт. Були випадки, коли ентузіасти змушують Doom 3 запускати на непідтримуваних графічних чіпах, таких як застарілий застарілий Voodoo 2, але вони не здатні відображати кожен піксельний підсвічування та відбиток.

Рейтинг: 7/10

2.3 Висновок

Розібравши всі сильні та слабкі сторони таких двигунів, як:

- Unity
- Unreal Engine 4
- Id Tech 4

, а також зробивши аналіз їх можливостей і вивівши рейтинг кожного з них, можна зробити висновок, що Unreal Engine 4 майже ідеально підходить для розробки гри. Єдиним серйозним недоліком є доволі високий поріг входу для нових програмістів та дизайнерів, проте це може бути усунене якісним курсом розробки.

3 Презентація створеної гри

3.1 Презентація вигляду гри

В даному розділі будуть представлені графічні матеріали (скріншоти), котрі розкриють вигляд та представлять розроблену гру, котру студенти зможуть почати розробляти після засвоєння курсу.

Рис. 3.1 - Початок гри

Рис. 3.2 - Презентація HUD

Рис. 3.3 - Презентація HUD

Рис. 3.4 - Меню паузи

Рис. 3.5 - Можливі опції

Рис. 3.6 - Процес виходу в головне меню

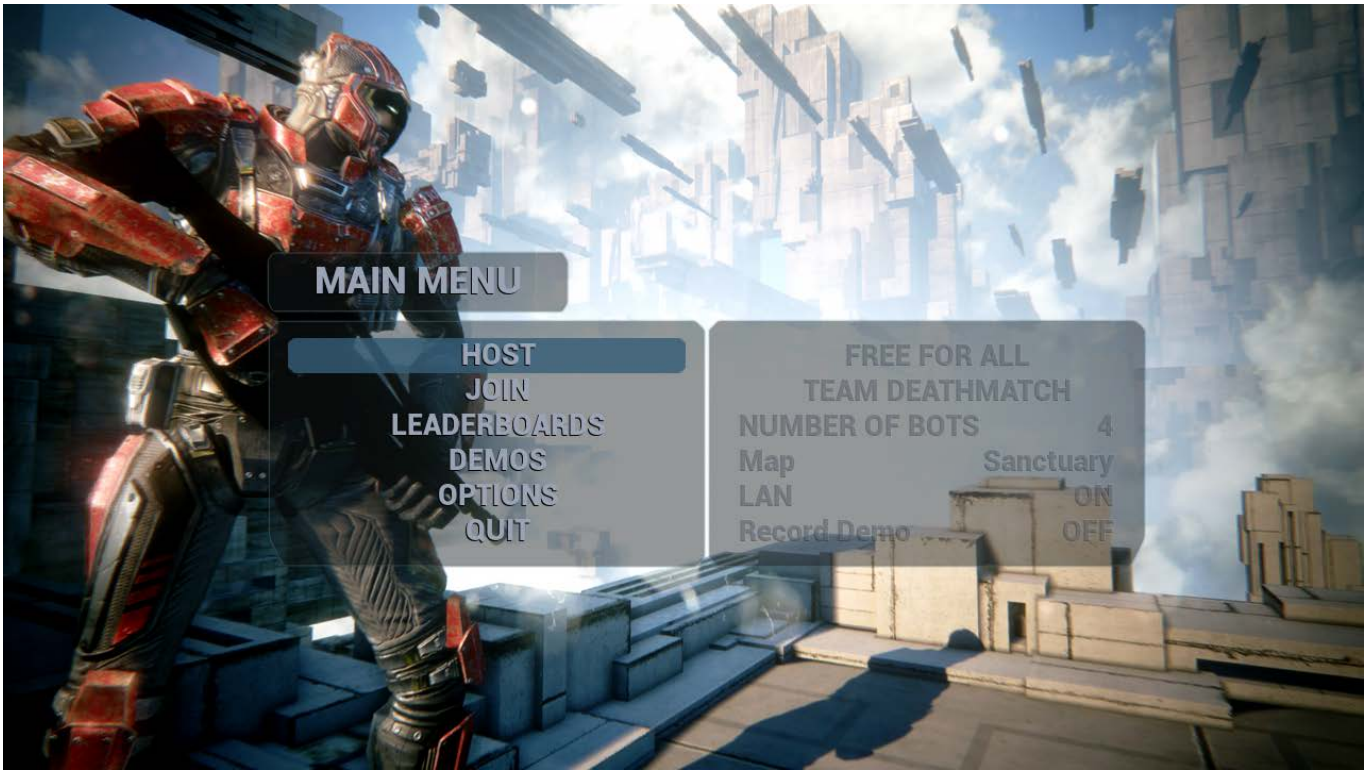

Рис. 3.7 - Головне меню

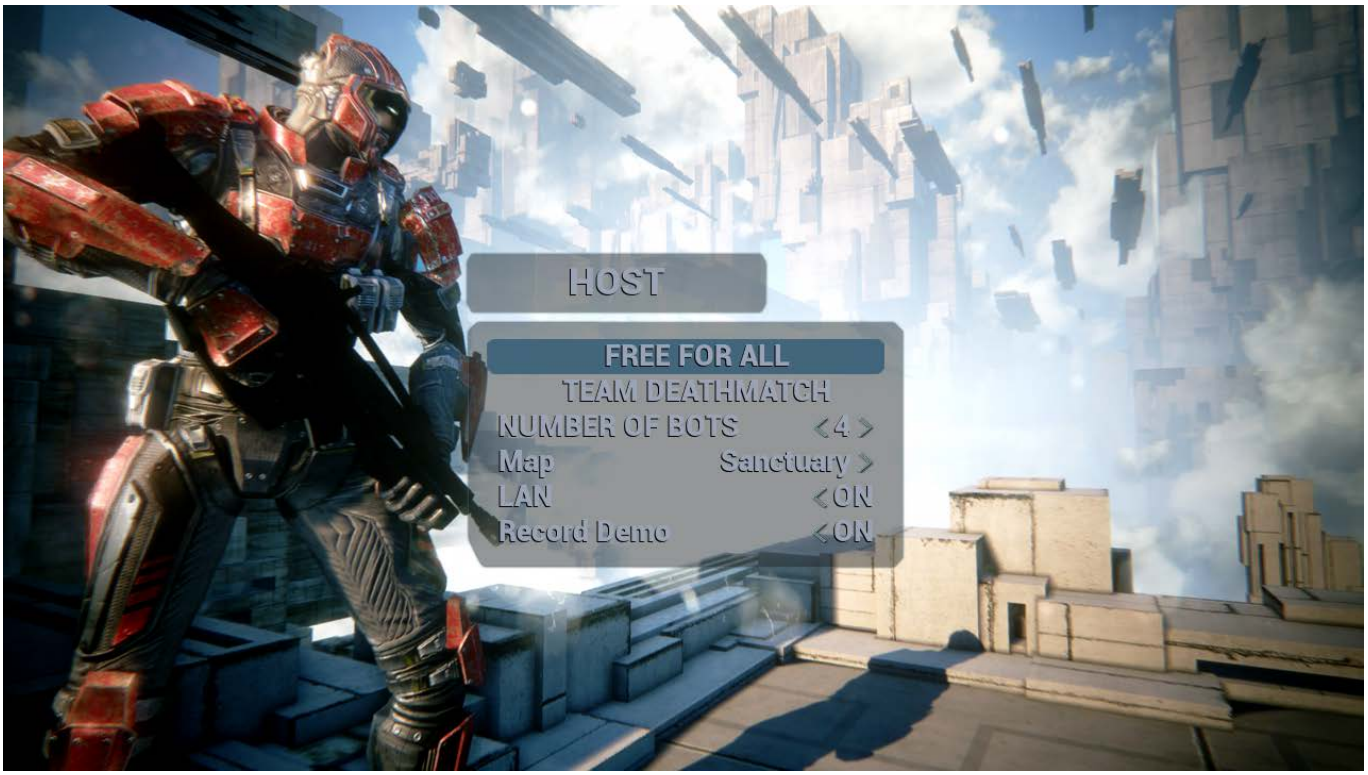

Рис. 3.8 - Створення мультиплеерної гри

Рис. 3.9 - Початок мультиплеерної гри

Рис. 3.10 - Результат смерті головного героя

Рис. 3.11 - Результат матчу

Рис. 3.12 - Вигляд одного з рівнів до початку гри

Рис. 3.13 - Гра на рівні Highrise

Рис. 3.14 - Взаємодія з ботами

3.2 Опис можливостей гри та використаних технологій

Створена гра була розроблена на Unreal Engine 4 використовуючи базові безкоштовні ассети, які були представлені ігровим двигуном.

Створена гра володіє такими характеристиками:

1. Переміщення (базове переміщення, стрибки, прискорення)
2. Стрільба
3. Декілька видів зброї (гвинтівка, гранатомет)
4. Два повноцінні рівні для боротьби
5. Можливість налаштування
   5.1. Роздільна здатність
   5.2. Якість
   5.3. Повноекранний режим
   5.4. Корекція гамми
   5.5. Чутливість прицілювання

6. Штучний інтелект
   6.1. Кожен сам за себе
   6.2. Команда на команду
7. Налаштування кількості ботів
8. Робота через мережу Іентернет
9. Можливість підключення по IP
10. Можливість запису гри

3.3 Висновок

В ході роботи було створено комп’ютерну гру, яка може бути охарактеризована, як шутер від першого лиця, з вбудованим штучним інтелектом, з можливістю гри використовуючи мережу Інтернет, з можливістю гри з іншими гравцями.

В даному розділі було представлено скріншоти різних моментів гри, котрі дозволяють в деякій мірі представити результат якого можуть добитися студенти вивчаючи розробку комп’ютерних ігор. Це тільки одна з можливостей, які відкриються перед студентами в ході вивчення курсу.

4 Курс “Розробка комп’ютерних ігор”

4.1 План курсу “Розробка комп’ютерних ігор”

- Урок №1: Початок
- Урок №2: Інтерфейс, імпорт та додавання мешів на рівень
- Урок №3: Матеріали
- Урок №4: Блупрінти
- Урок №5: Створення GameMode
- Урок №6: Створення персонажу
- Урок №8: Контроль камери
- Урок №9: Стрибки
- Урок №10: Меш для персонажа
- Урок №11: Зміна виду камери
- Урок №12: Додавання мешу для виду від першої особи
- Урок №13: Стрільба та пулі
- Урок №14: Коллізія снаряду та час життя
- Урок №15: Взаємодія снаряду з об’єктами

4.2 Урок №1: Початок

Unreal Engine 4 - це набір інструментів для розробки будь-яких за складністю ігор, починаючи від простих 2D до AAA-проектів. На цьому двигуні розроблені такі ігри, як ARK: Survival Evolved, Tekken 7 і Kingdom Hearts III.

Розробка в Unreal Engine 4 дуже проста навіть для початківців. Використовуючи систему Visual Scripting Blueprints, ви можете створювати повномасштабні ігри без написання жодного рядка коду! Поєднуючи простий у використанні інтерфейс, ви можете швидко отримати прототип і запустити його.

Наш курс з Unreal Engine 4 спрямований на допомогу початківцям. Ось основні моменти, які буде розглянуто в цьому уроку:

- Встановлення двигуна
- Імпорт активів
- Створення матеріалів
- Використання Blueprints для створення об'єктів із базовою функціональністю

Щоб дізнатись про це, ви створите рухому платформу, на якій стоїть банан.

4.2.1 Встановлення Unreal Engine 4

Для встановлення Unreal Engine 4 використовується програма Epic Games Launcher. Перейдіть на веб-сайт Unreal Engine та натисніть кнопку "Get Unreal" у

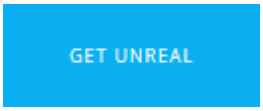

Рис. 4.1- Get Unreal

верхньому правому куті.

Вам потрібно буде створити обліковий запис, перш ніж почати запуск. Після створення облікового запису завантажте панель запуску для своєї операційної системи.

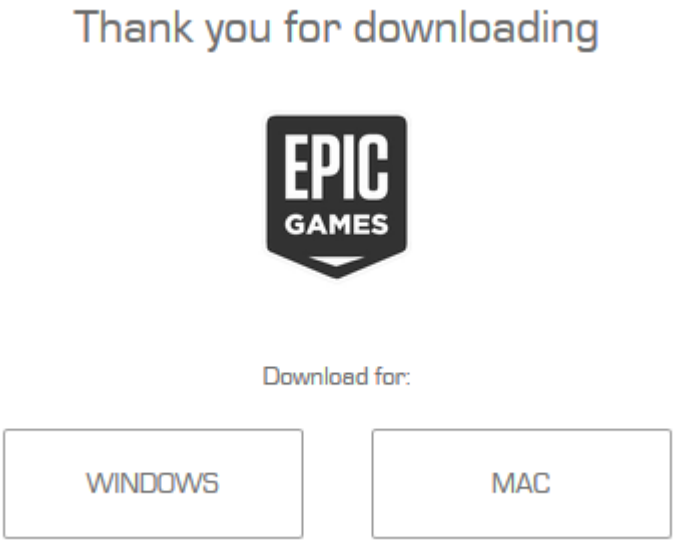

Рис. 4.2 - Вибір типу завантаження

Завантаживши та встановивши панель запуску, відкрийте її. З'явиться таке вікно:

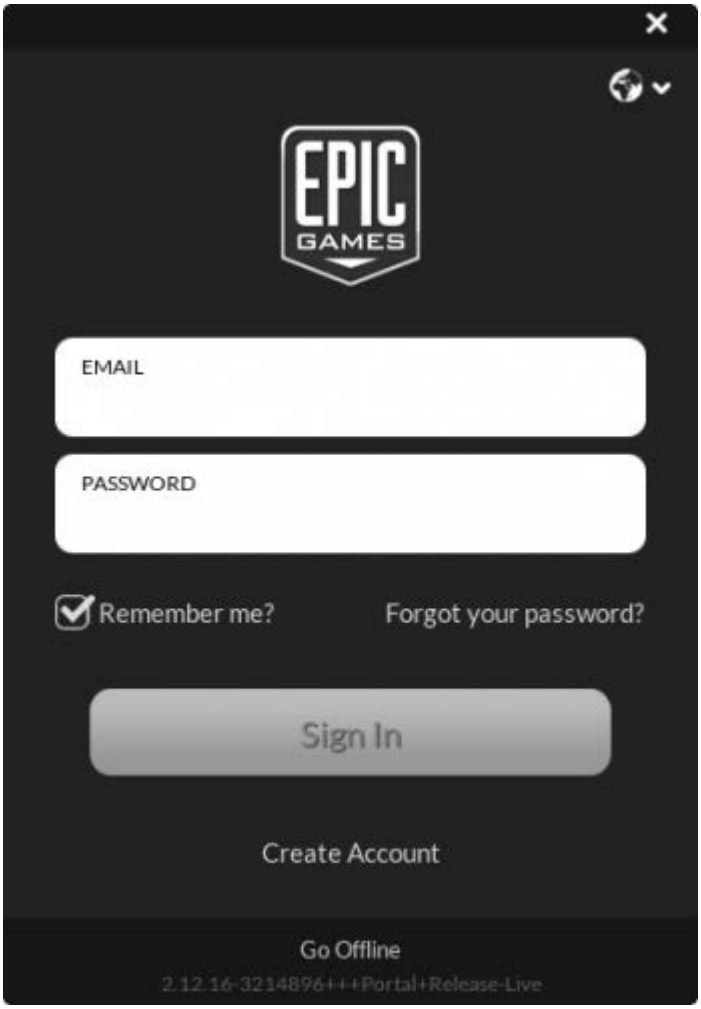

Рис. 4.3 - Вікно аутентифікації

Введіть електронну адресу та пароль, які ви використовували для завантаження панелі запуску, та натисніть “Увійти”. Після входу з'явиться таке вікно:

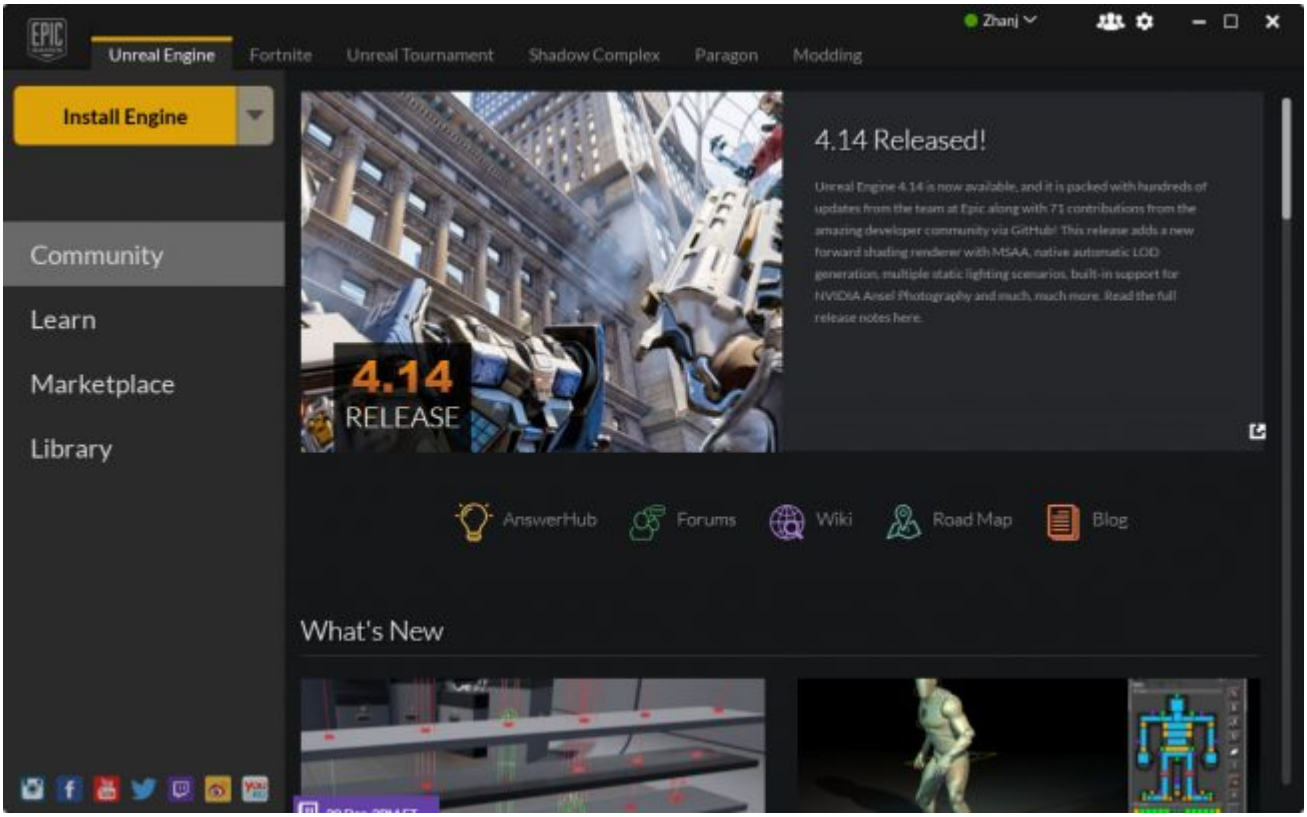

Рис. 4.4 - Головне вікно Epic Games

Введіть адресу електронної пошти та пароль у верхньому лівому куті, натисніть кнопку “Встановити двигун”. Панель виведе вас на екран, де ви можете обрати, які компоненти потрібно встановити.

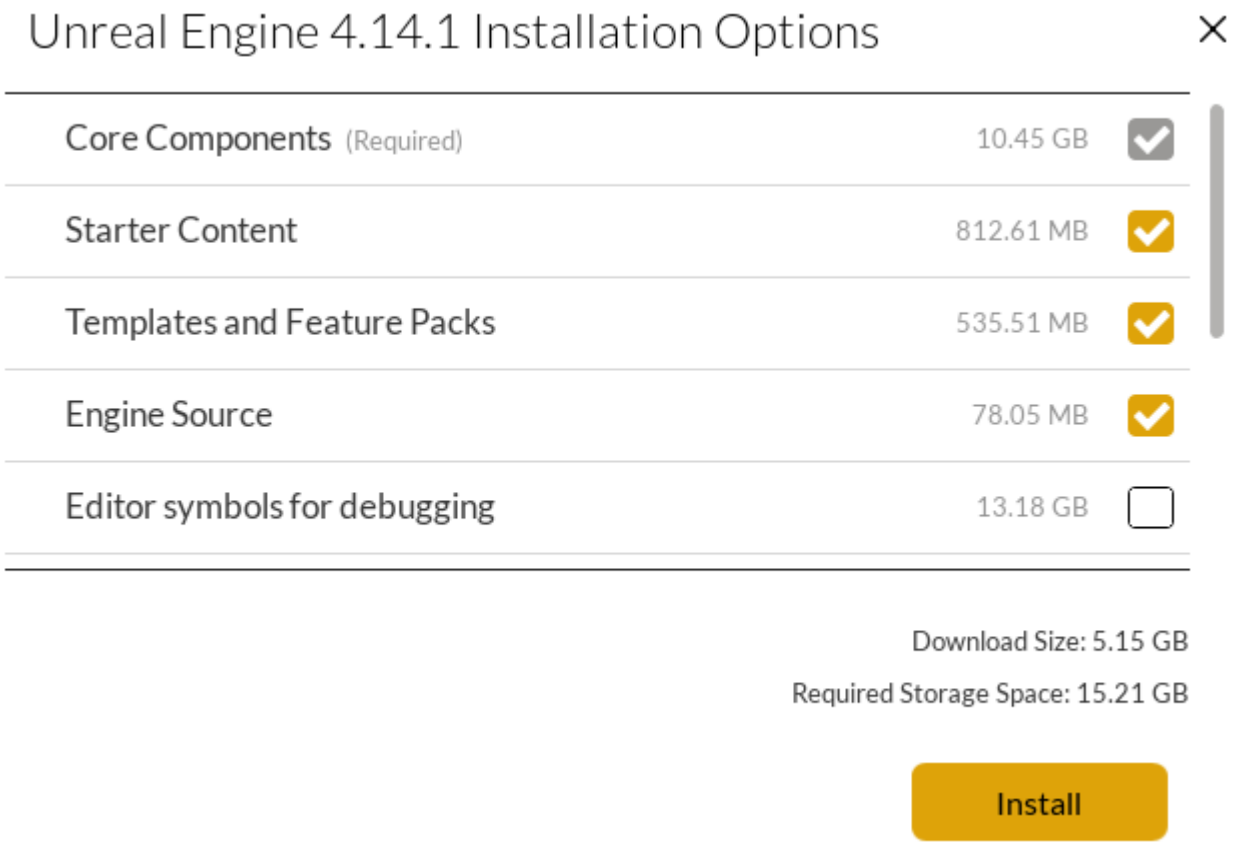

Рис. 4.5 - Опції встановлення 1

За замовчуванням встановлюється: “Контент для початківців”, “Шаблони та Пакети функцій” та “Джерело двигуна”. Краще всього залишити усе це відміченим через те, що:

- Контент для початківців. Це низка ресурсів, які ви можете безкоштовно використовувати у своїх проектах. Там можна знайти певні моделі та матеріали. Ви можете використовувати ці ресурси як плейсхолдери або в релізній версії.
- Шаблони та компоненти функцій: шаблони встановлюють основні функціональні можливості, що стосуються обраного вами жанру. Наприклад, підбираючи шаблон Side Scroller, буде створений проект із символом, основним рухом і фіксованою плоскою камерою.
- Джерело двигуна від Epic забезпечує доступ до вихідного коду, тобто кожен може вносити зміни до двигуна. Наприклад, якщо ви хочете

додати користувацькі функції до редактора, ви можете це зробити, змінивши вихідний код.

Переглянувши список, ви побачите, що доступні різні платформи. Якщо ви не плануєте розробляти певну платформу, можете сміливо її відключити.

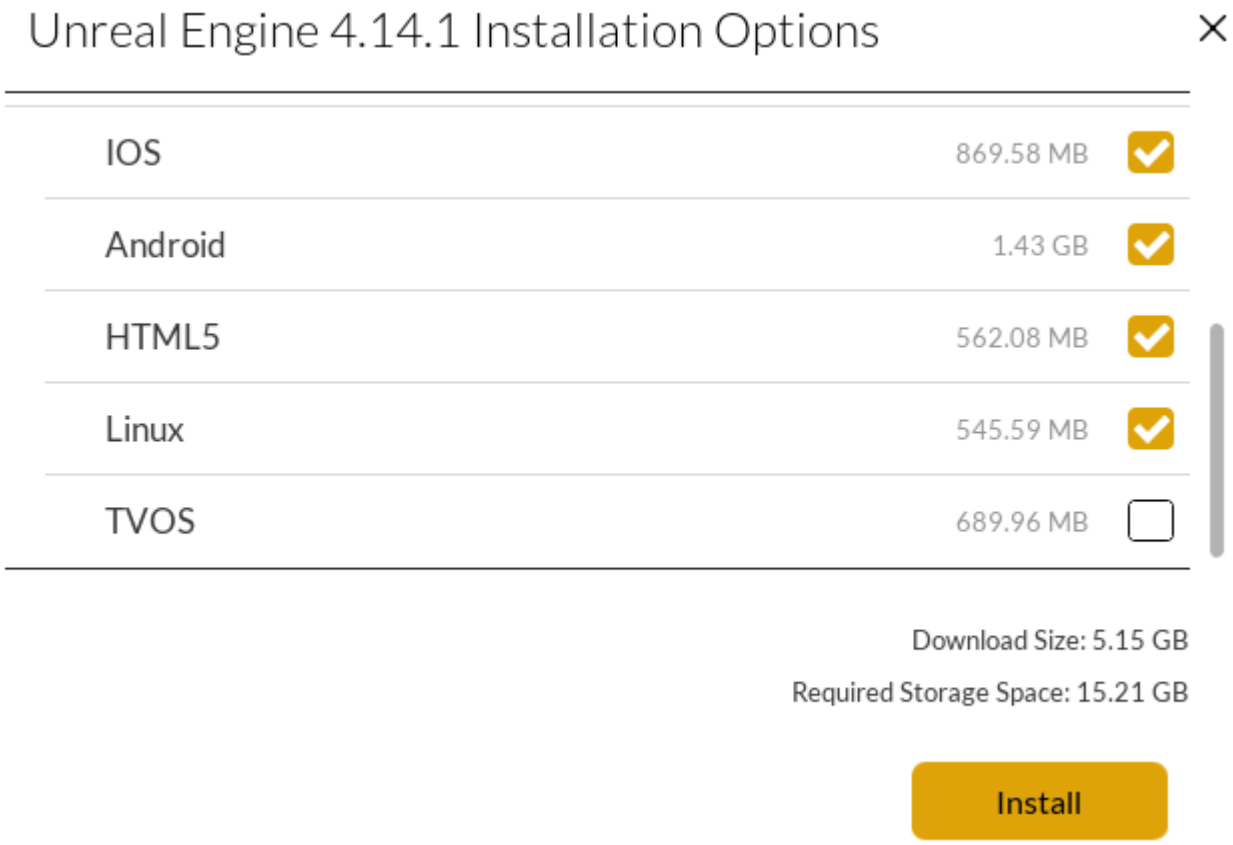

Рис. 4.6 - Опції встановлення 2

Вибравши компоненти, натисніть кнопку “Встановити”. Після завершення встановлення двигун з'явиться у вашій бібліотеці. Настав час створювати проект.

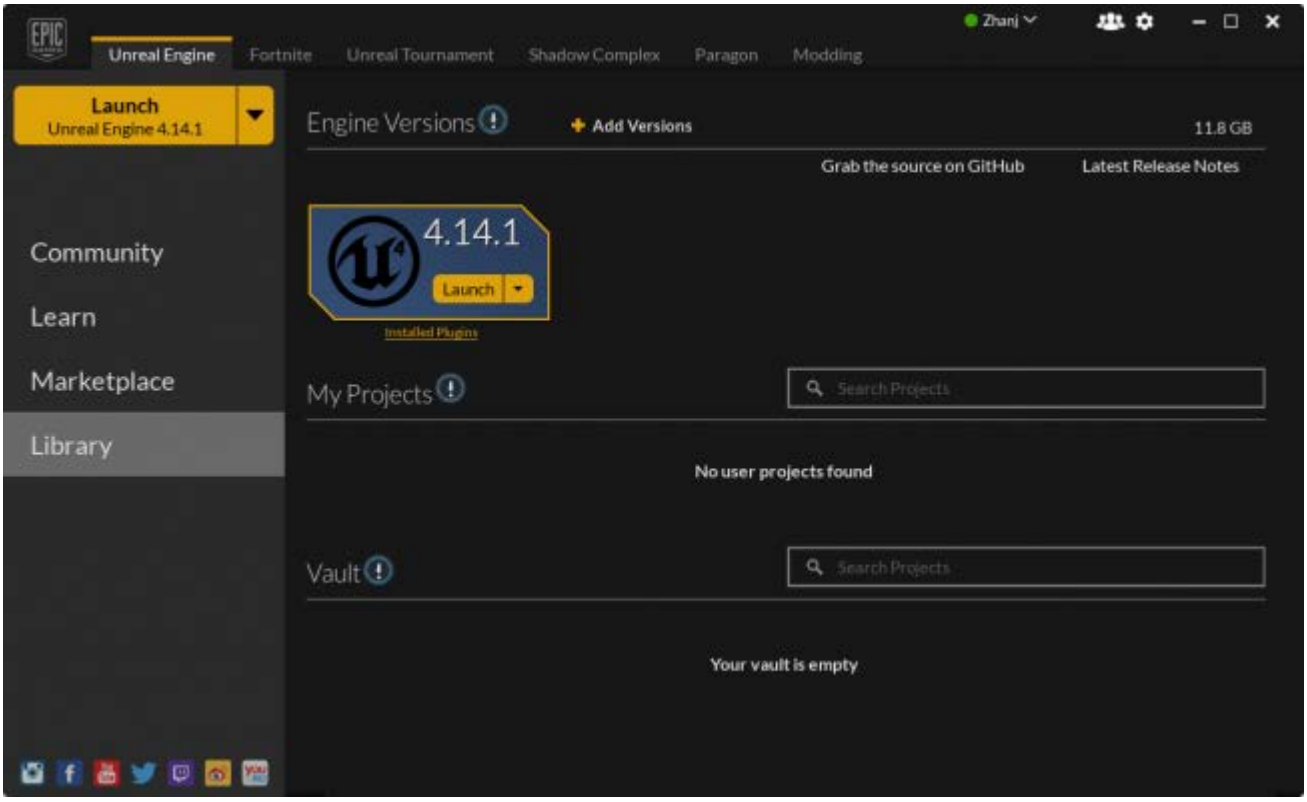

Рис. 4.7 - Бібліотека Epic Games

4.2.2 Створення проекту

Натисніть одну з кнопок запуску, щоб відкрити браузер проекту. Коли він відкриється, перейдіть на вкладку “Новий проект”.

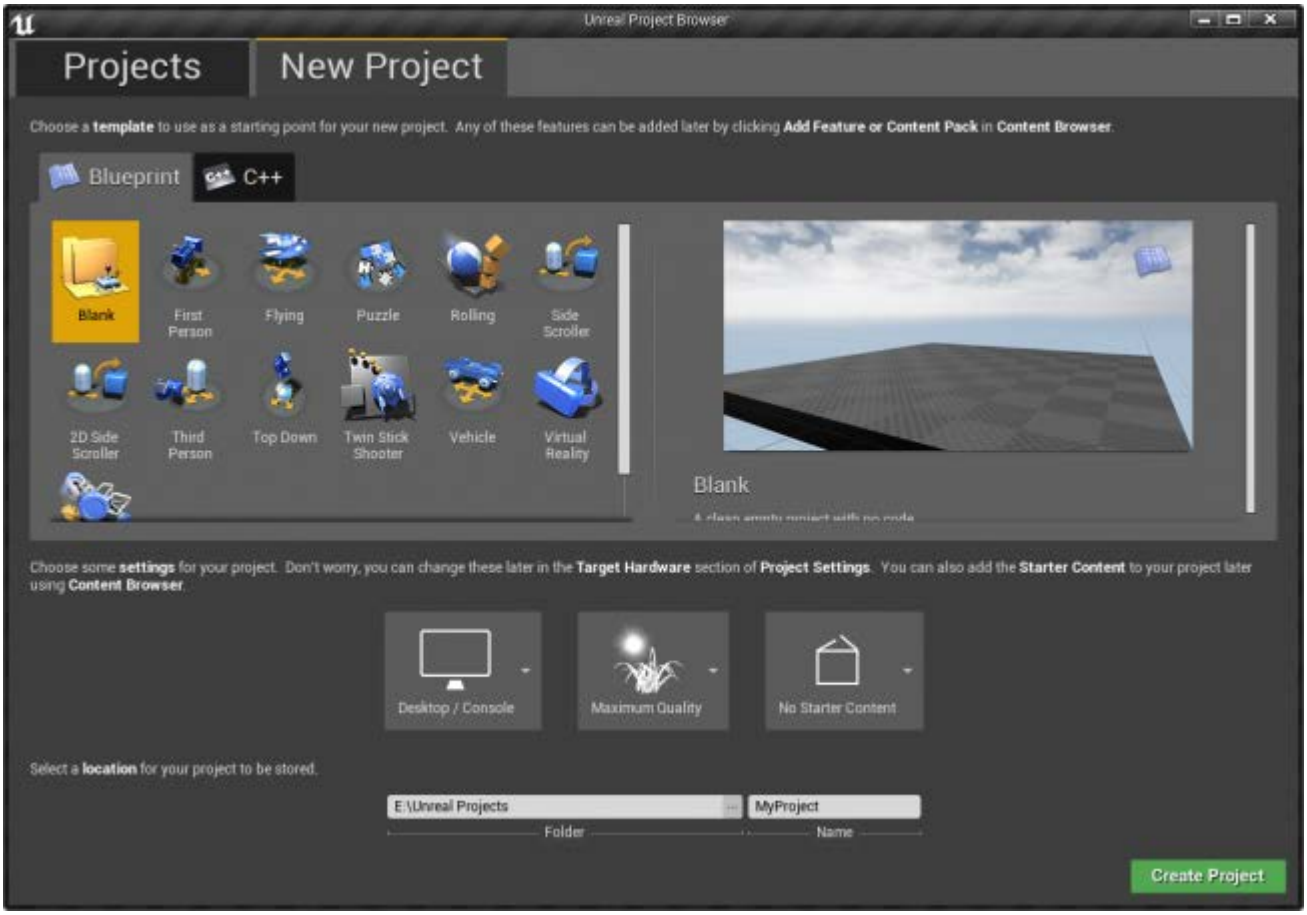

Рис. 4.8 - Створення проекту

Перейдіть на вкладку “Бланкет”. Тут ви можете використовувати один із шаблонів. Однак, оскільки ви починаєте з нуля, виберіть шаблон “Blank”.

Нижче ви знайдете додаткові налаштування.

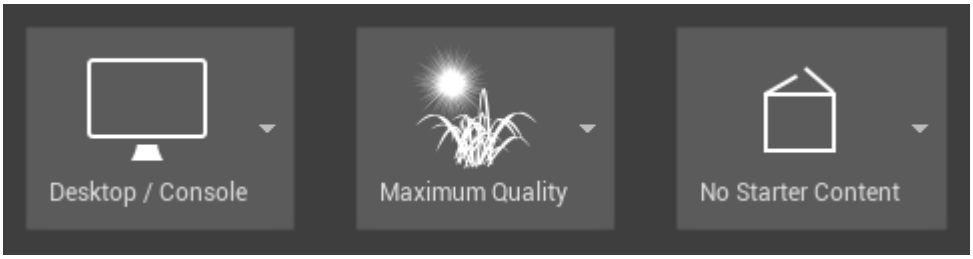

Рис. 4.9 - Вибір базових налаштувань

Ось що робить кожен варіант:

- Цільове обладнання: під час вибору мобільних пристроїв / планшетів буде вимкнено деякі ефекти після обробки. Це також дозволить використовувати мишу як сенсорний вхід. Встановіть це на Desktop / Console.

- Графічна ціль: під час вибору масштабованого 3D або 2D буде вимкнено деякі ефекти після обробки. Установіть це значення на максимальну якість.
- Контент для початківців. Ви можете включити цю опцію, щоб включити контент для початківців – це додасть до гри базовий контент, за допомогою якого можна отримати базову модель гри.

До того ж, є розділ, в якому вказується розташування вашої папки проекту та ім'я вашого проекту.

Рис. 4.10 - Вибір розташування

Ви можете змінити розташування папки проекту, натиснувши три крапки в кінці поля папки.

Назва проекту не відображає назву гри, тому не хвилюйтеся, якщо ви захочете змінити назву пізніше. Виділіть текст у полі “Ім'я” і введіть BananaTurntable.

Нарешті, натисніть кнопку “Створити проект”.

4.3 Урок №2: Інтерфейс, імпорт та додавання мешів на рівень

4.3.1 Навігація в інтерфейсі

Після створення проекту відкриється редактор. Редактор розділений на кілька панелей:

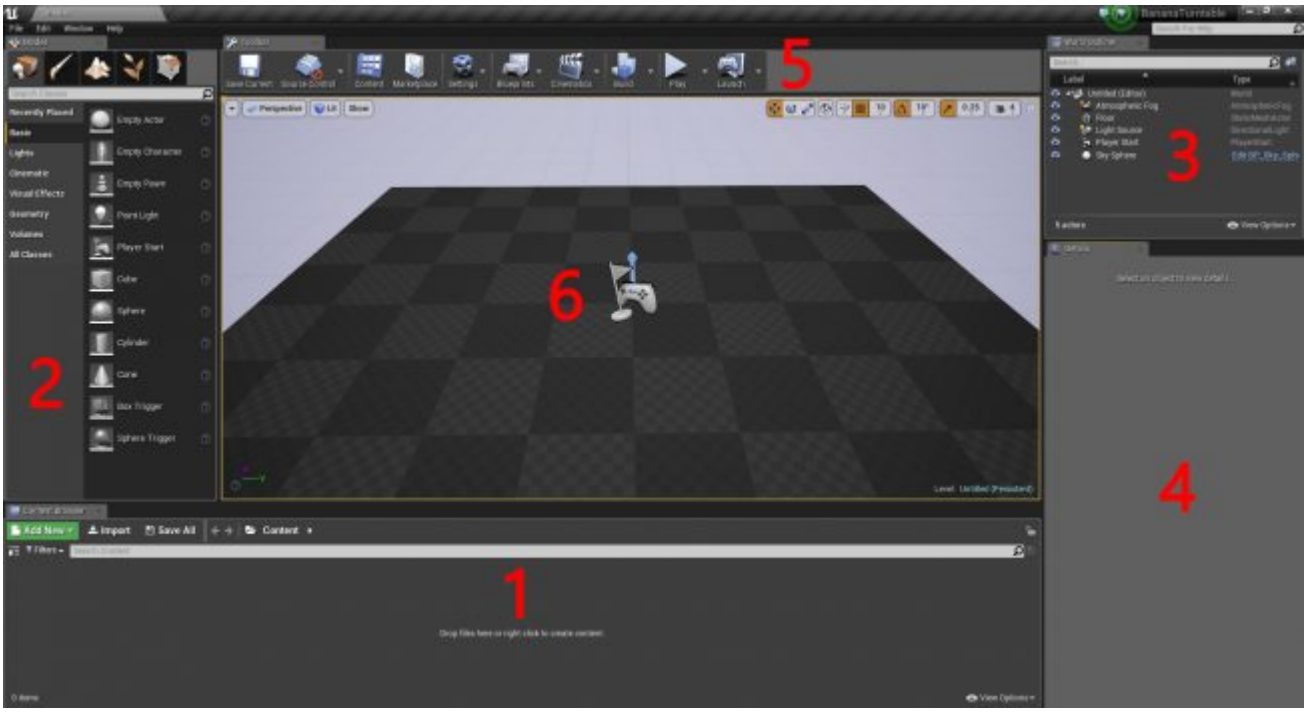

Рис. 4.11 - Базова наівігація

- Браузер контенту: на цій панелі відображаються всі файли проекту. Використовуйте її для створення папок і упорядкування ваших файлів. Ви можете шукати свої файли за допомогою панелі пошуку або за допомогою фільтрів.
- Режими: на цій панелі можна вибрати такі інструменти, як “Ландшафтний інструмент” та “Інструмент Foliage”. Інструмент “Місце” є інструментом за замовчуванням. Це дозволяє розміщувати на вашому рівні багато різних типів об'єктів, таких як ліхтарі та камери.
- World Outliner: Відображає всі об'єкти на поточному рівні. Ви можете організувати список, поставивши відповідні елементи в папки. Також є можливість пошуку та фільтрування за типом.

- Подробиці: будь-який об'єкт, який ви виберете, матиме тут свої властивості. Використовуйте цю панель для редагування параметрів об'єкта. Зроблені зміни вплинуть лише на обраний екземпляр об'єкта. Наприклад, якщо у вас є дві сфери та ви зміните розмір однієї, це вплине лише на даний об'єкт.
- Панель інструментів: містить багато різних функцій. Більш за все вам знадобиться Play.
- Viewport: це вигляд вашого рівня. Ви можете “озирнутися” натиском правої кнопки та переміщенням миші. Щоб перемістити камеру, клацніть правою кнопкою миші та використовуйте клавіші WASD.

4.3.2 Імпортування ассетів

Який сенс у тому, щоб мати рухомий стіл, на якому нічого не відображається? Завантажте цю модель банану. Усередині є два файли: Banana_Model.fbx і Banana_Texture.jpg. Як альтернатива, ви можете використовувати вашу власну модель.

Перш ніж Unreal зможе використовувати будь-які файли, їх потрібно імпортувати. Перейдіть до браузера змісту та натисніть “Імпортувати”.

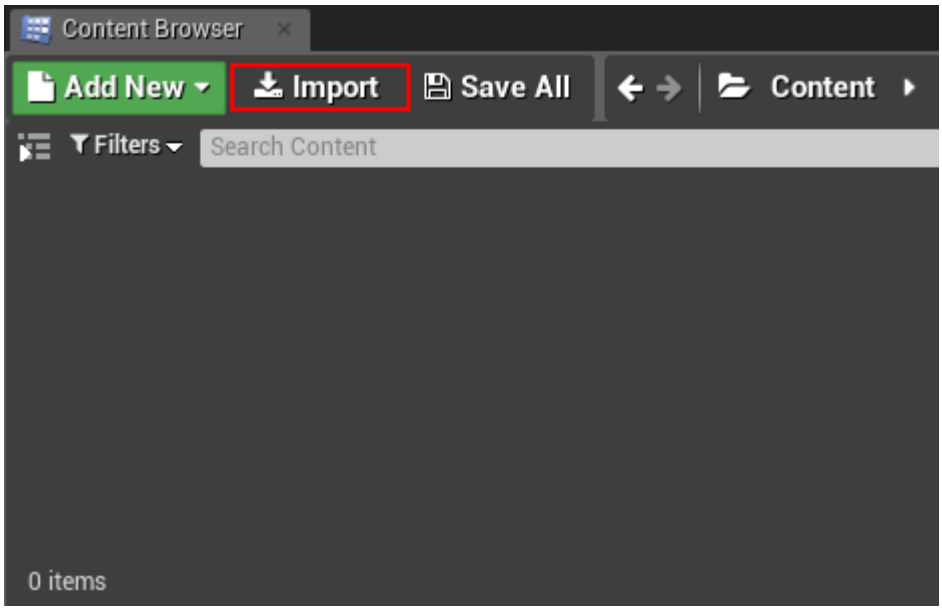

Рис. 4.12 - Імпорт контенту

Використовуючи файловий браузер, знайдіть папку, де є Banana_Model.fbx та Banana_Texture.jpg. Перетягніть, оберіть обидва файли та натисніть “Відкрити”.

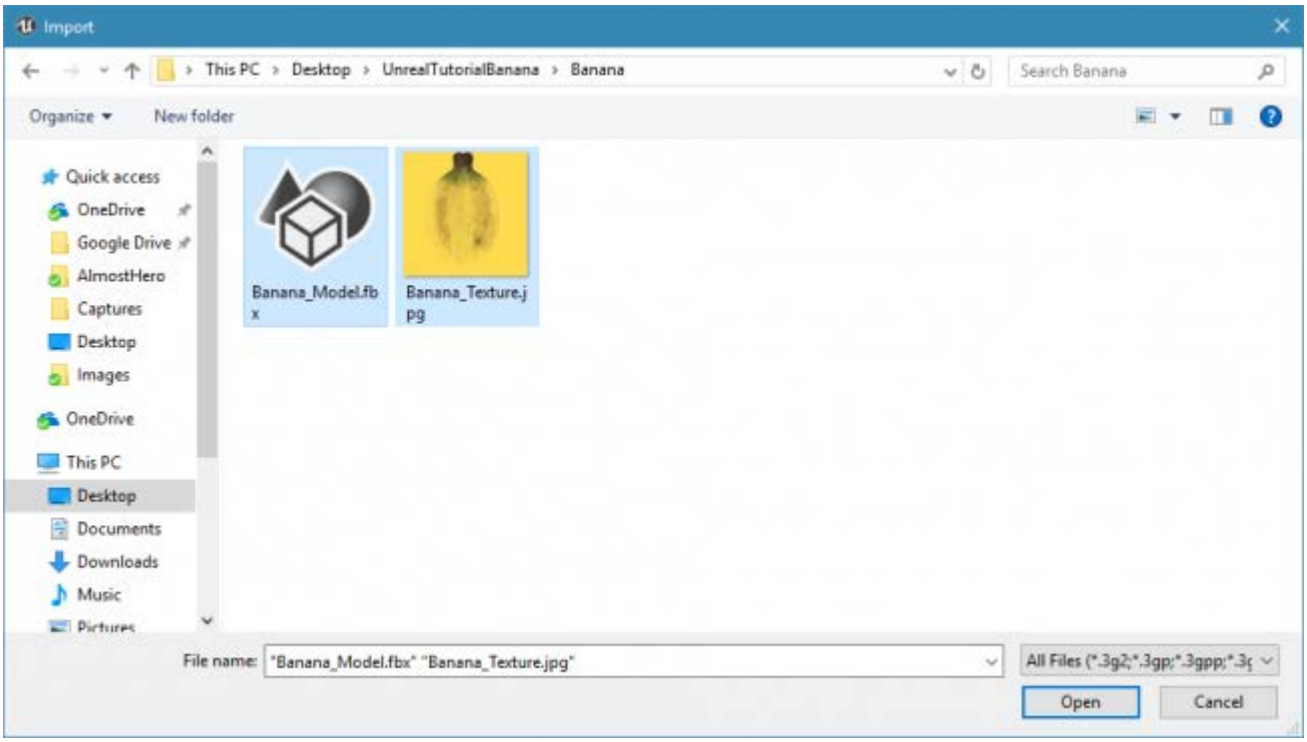

Рис. 4.13 - Імпорт ассету

Unreal дасть вам деякі параметри імпорту для файлу .fbx. Переконайтеся, що “Імпортувати Матеріали” не буде позначено, оскільки ви будете створювати власні матеріали. Ви можете зробити інші налаштування окремо.

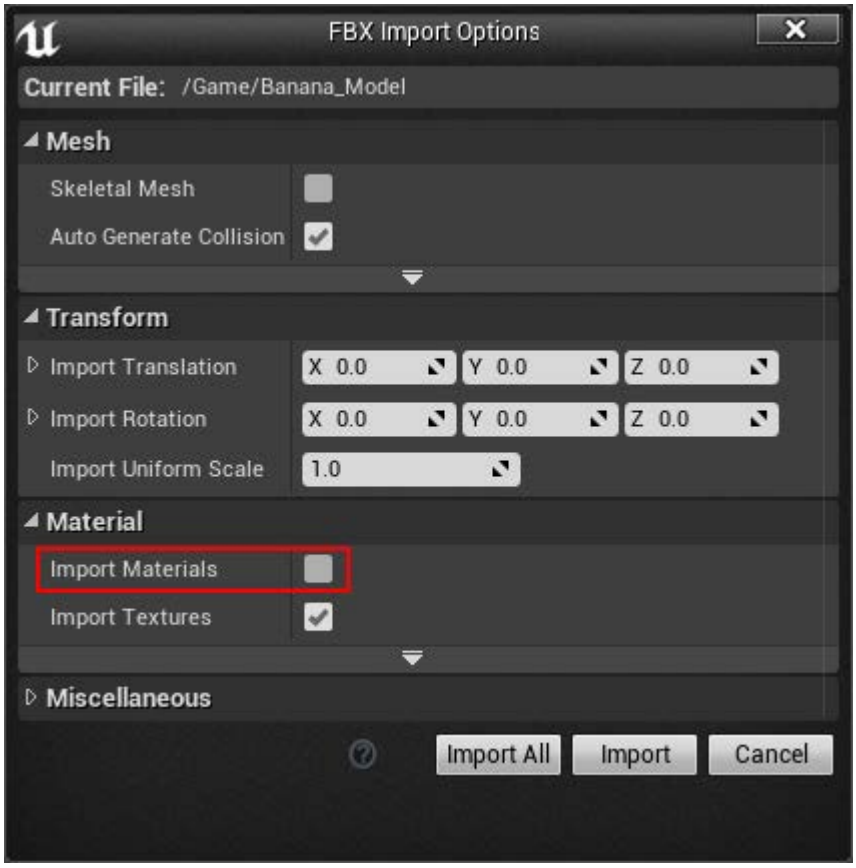

Рис. 4.14 - Імпорт матеріалів

Натисніть “Імпортувати”. Тепер два файли з'являться у вашому веб-переглядачі контенту.

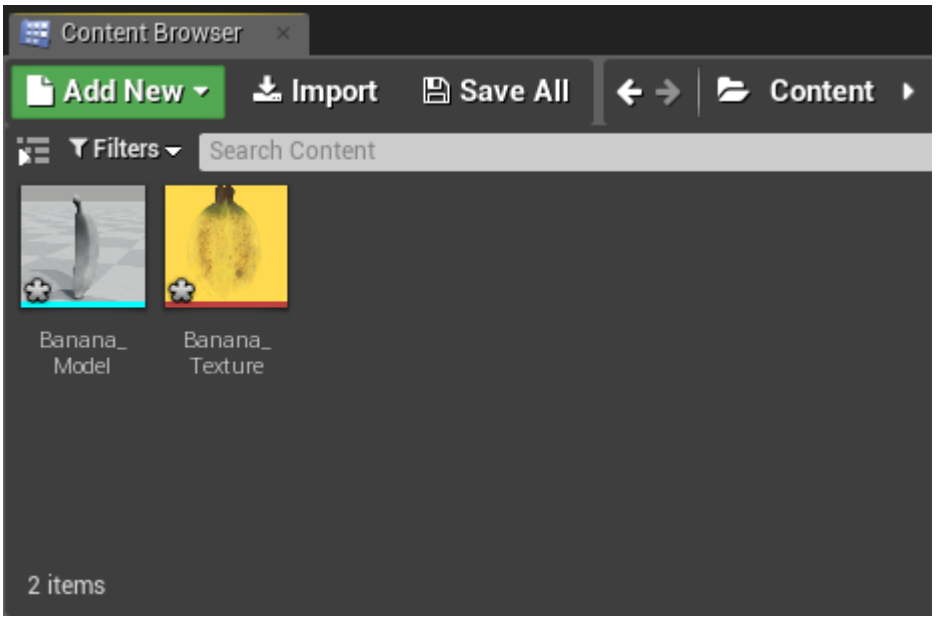

Рис. 4.15 - Контент браузер

Коли ви імпортуєте файл, воно фактично не зберігається у вашому проекті, доки ви це особисто не зробите. Ви можете зберегти файли, клацнувши на файл правою кнопкою миші та вибравши “Зберегти”. Ви також можете одночасно зберігати всі файли, вибравши File/Save All.

Зауважте, що моделі в Unreal називаються сітками. Отже, тепер, коли у вас є сітка для вашого банану, час помістити його в рівень.

4.3.3 Додавання мешів на рівень

Рівень виглядає дуже порожньо на даний момент.

Щоб додати сітку до рівня, клацніть лівою кнопкою миші та перетягніть Banana_Model з веб-переглядача контенту в Viewport. Відпускання лівої клавіші миші поставить сітку.

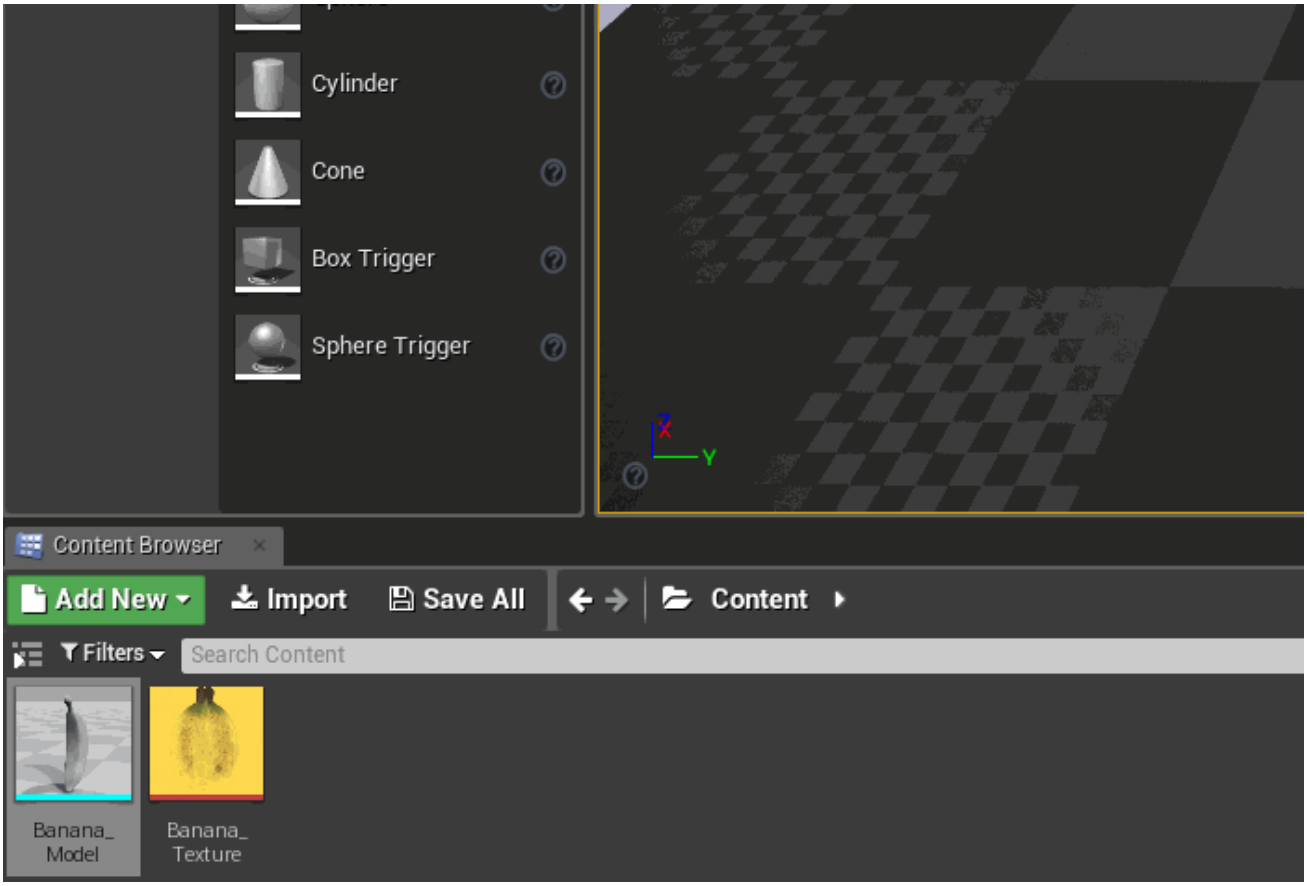

Рис. 4.16 - Додання об'єкту із контент браузеру

Об'єкти на рівні можна переміщувати, обертати та масштабувати. Комбінації клавіш для них - W, E та R. Ви можете використовувати таким чином маніпулятор:

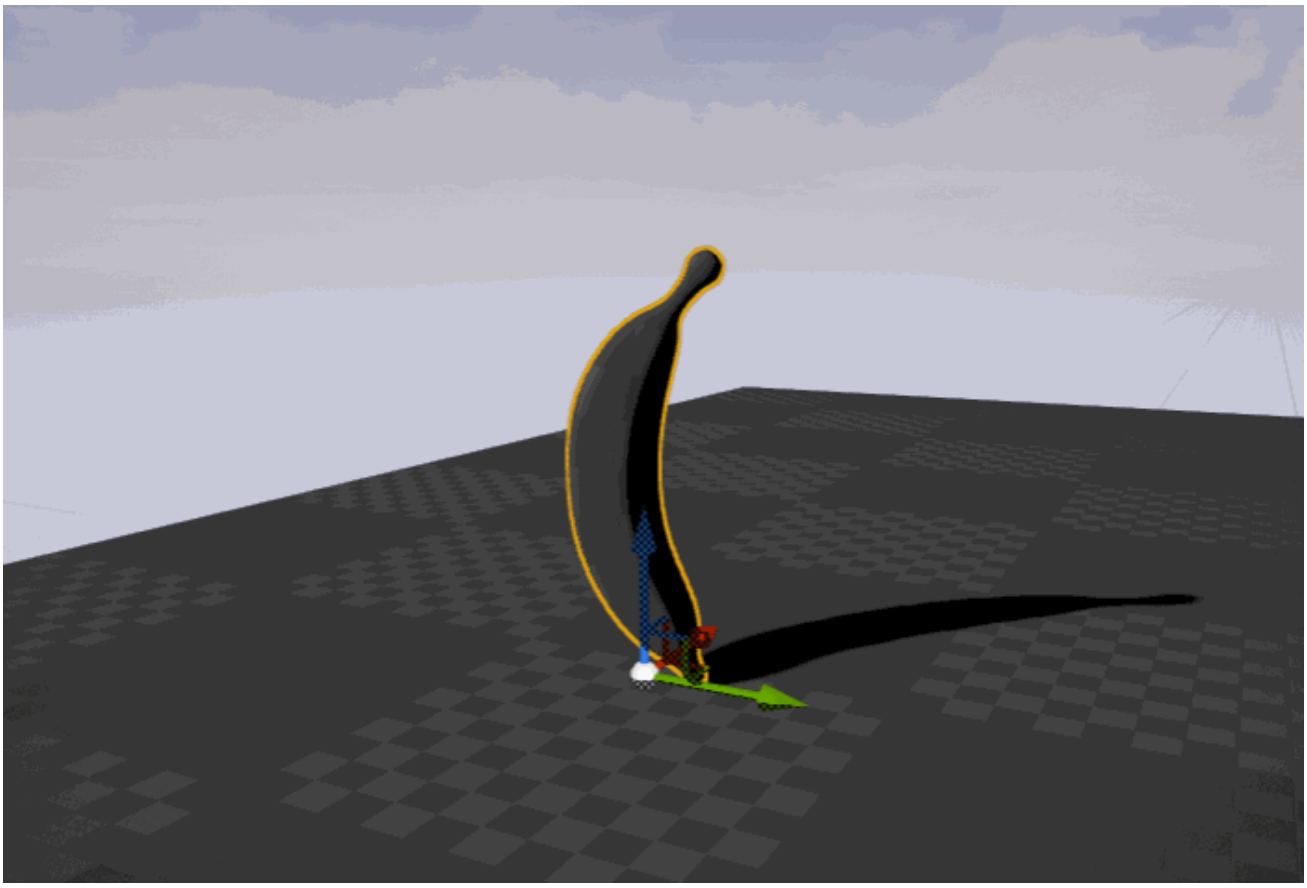

Рис. 4.17 - Об'єкт на рівні

4.4 Урок №3: Матеріали

4.4.1 Про матеріали

Якщо ви уважно подивитеся на банан, ви побачите, що він взагалі не жовтий! Насправді, він майже сірий.

Щоб додати банану якийсь колір і деталі, потрібно створити матеріал.

Що таке Матеріал?

Матеріал визначає, як виглядає поверхня. На базовому рівні матеріал впливає на чотири характеристики:

- Колір основи (Base Color): колір або текстура поверхні. Використовується для додавання деталей та варіантів кольорів.
- Металевість (Metallic): як “металева” поверхня. Як правило, чистий метал буде мати максимально металеве значення, тоді як тканина буде мати значення нуля.
- Блиск (Specular): контролює блиск нерудних поверхонь. Наприклад, кераміка матиме високе значення, але глина - навпаки.
- Шороховатості (Roughness): поверхня з максимальною шорсткістю не матиме ніякої блискучості. Використовується для поверхонь, таких як камінь та дерево.

Нижче наведено приклад трьох різних матеріалів. Вони мають однаковий колір, але різні атрибути. Інші атрибути встановлюються в нуль.

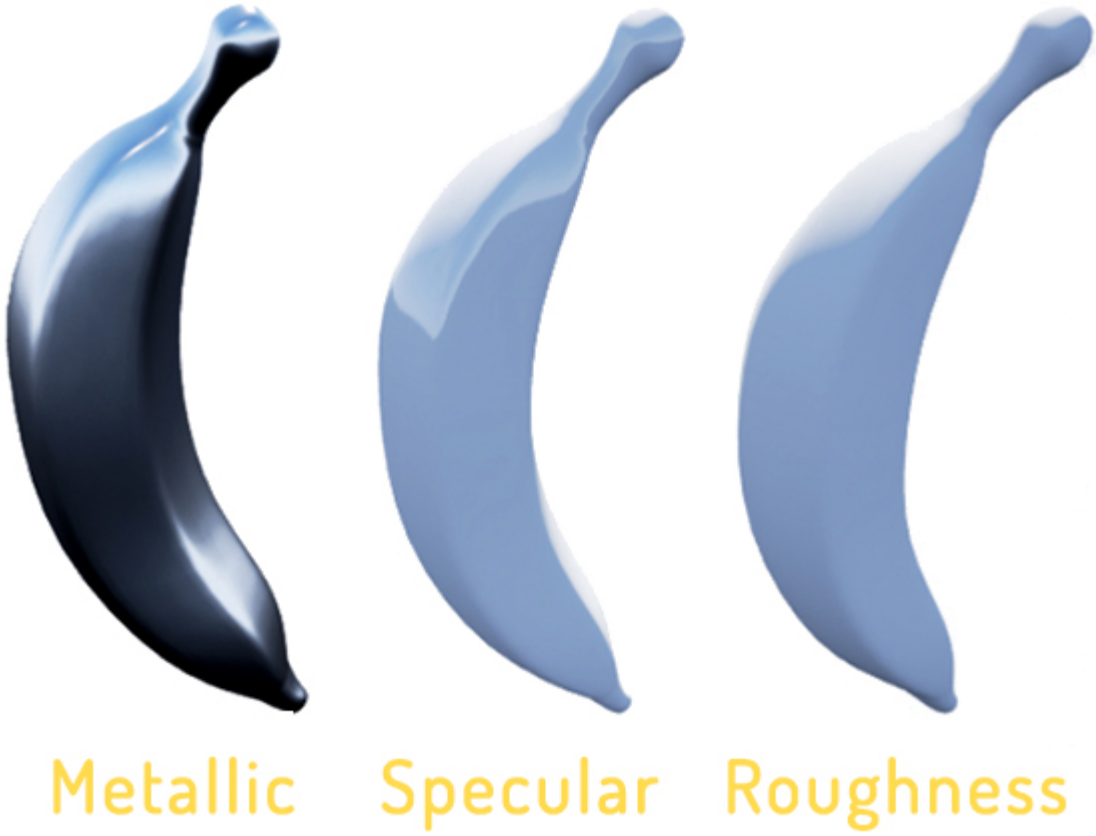

Рис. 4.18 - Вигляд матеріалів

Створення матеріалу

Щоб створити матеріал, перейдіть до свого браузера контенту та натисніть зелену кнопку “Додати новий”. У меню з'явиться список активів, які ви можете створити. Натисніть “Матеріал”.

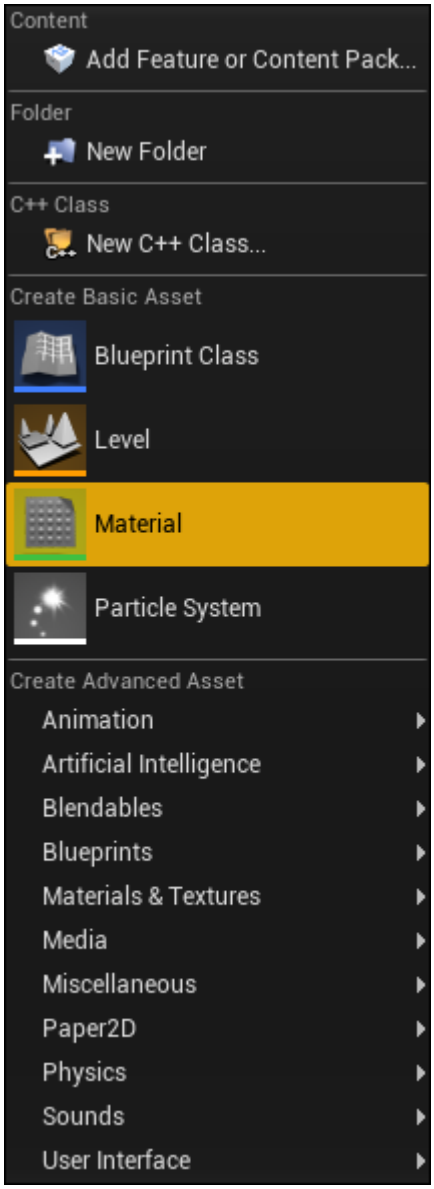

Рис. 4.19 - Додання матеріалів

Назвіть матеріал Banana_Material, а потім двічі клацніть файл, щоб відкрити його в редакторі матеріалу.

Рис. 4.20 - Вибір матеріалу

Матеріал редактора

Редактор матеріалів складається з п'яти основних панелей:

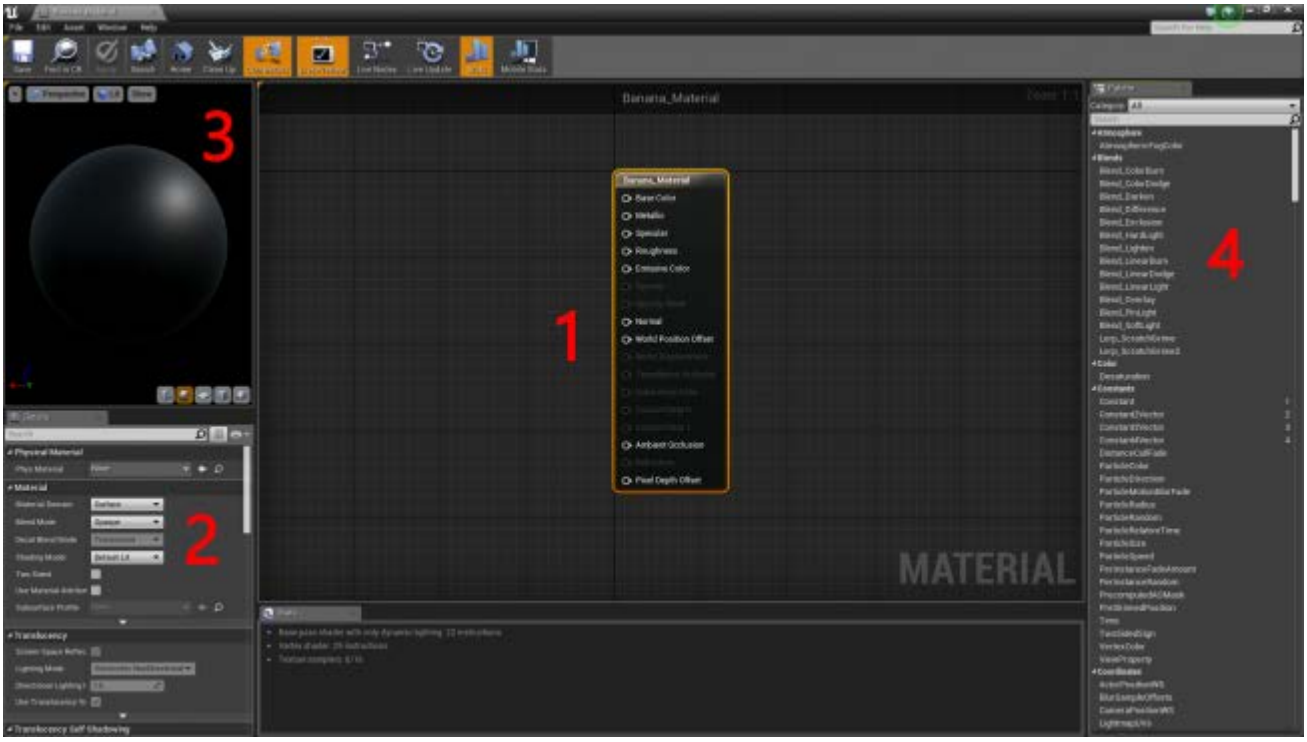

Рис. 4.21 - Використання матеріалів

- Графік: ця панель буде містити всі ваші вузли та вузол результатів. Збільшити можна, прокрутивши колесо миші.
- Подробиці: будь-який вибраний вами вузол буде відображатись тут. Якщо вузол не вибрано, на панелі відображатимуться властивості матеріалу.
- Viewport: Містить попередній перегляд сітки, в якому відображатиметься ваш матеріал. Оберніть камеру, утримуючи ліву кнопку миші та рухаючи її. Збільшити можна, прокрутивши колесо миші.
- Палітра: список усіх вузлів, доступних для вашого матеріалу.

Що таке вузол?

Перш ніж почати створювати матеріал, вам потрібно знати про об'єкти, які використовуються для його створення: вузли.

Вузли складають більшість матеріалу. Багато типів вузлів доступні для використання, пропонуючи різні функції.

Вузли можуть мати входи та виходи, представлені кружком зі стрілкою. Входи будуть розташовані з лівого боку, а виходи з'являться з правого боку.

Ось приклад, використовуючи вузол Multiply та Constant3Vector для додавання жовтого кольору до текстури:

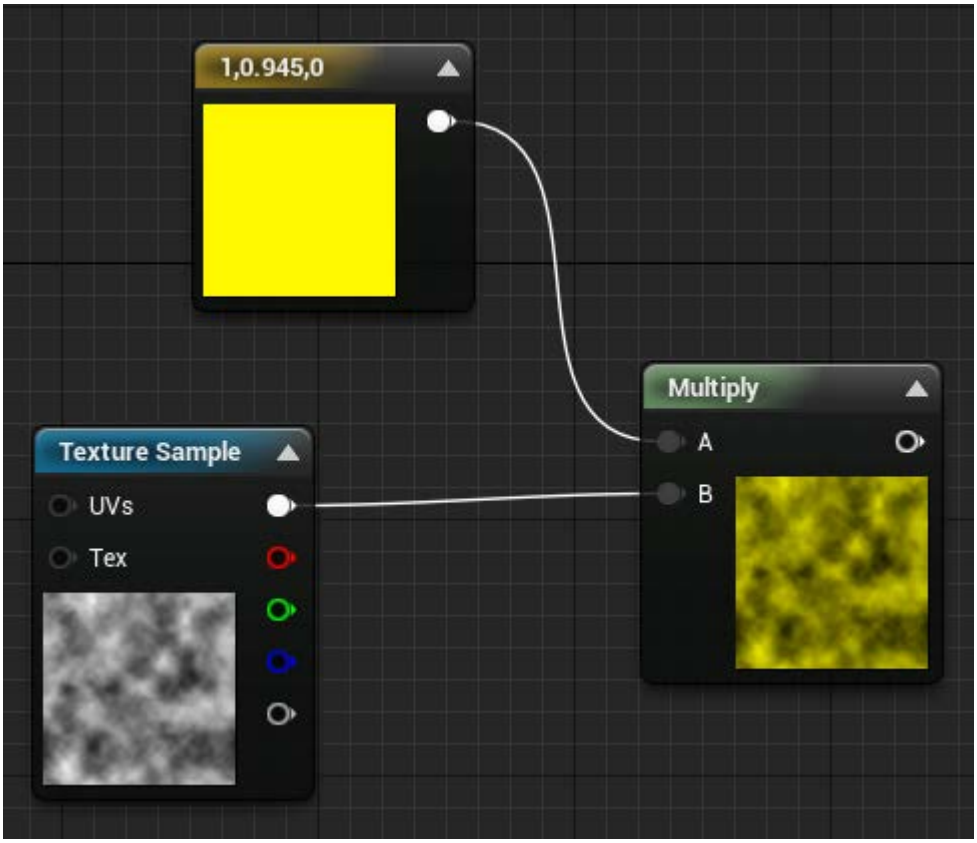

Рис. 4.22 - Використання текстур та матеріалів

Матеріали мають спеціальний вузол, який називається вузлом результатів, який вже був створений для вас у цьому випадку як Banana_Material. Що б ви не підключили до цього вузла, він визначатиме, як виглядає остаточний матеріал.

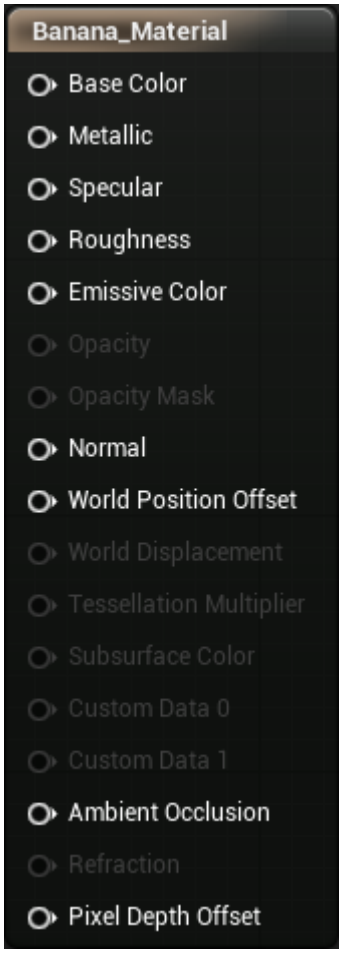

Рис. 4.23 - Блок матеріалу

Додавання текстур

Щоб додати колір та деталі до моделі, потрібна текстура. Текстури - це лише 2D зображення. Як правило, вони проектуються на 3D-моделі, щоб надати їм колір та деталі.

Для текстури банану ви будете використовувати Banana_Texture.jpg. Вузел TextureSample дозволить вам використовувати текстуру у вашому матеріалі.

Перейдіть на панель “Палітри” та знайдіть TextureSample. Додати вузол можна, утримуючи ліву кнопку та перетягнувши його на графік.

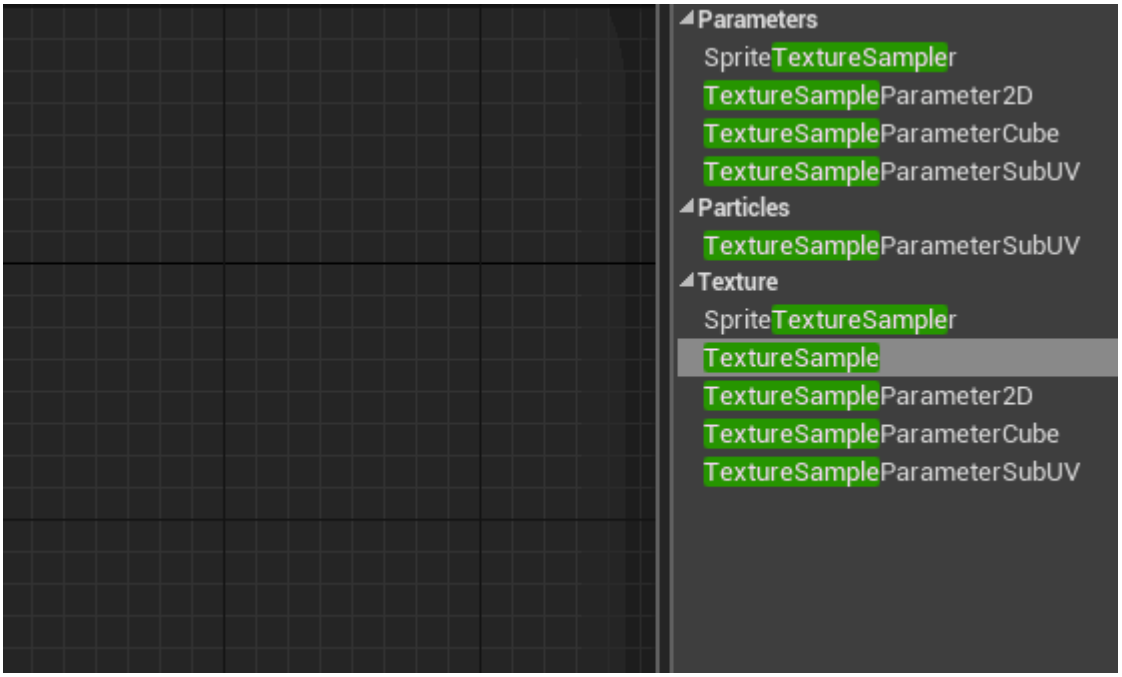

Рис. 4.24 - Додання текстури

Щоб вибрати текстуру, переконайтеся, що ви вибрали вузол TextureSample. Перейдіть на панель “Інформація” та натисніть меню, розташоване праворуч від текстури.

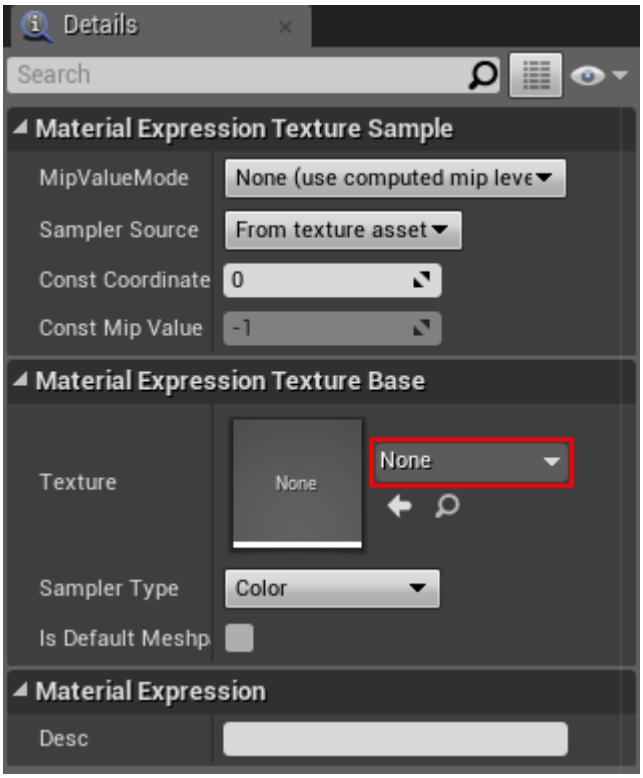

Рис. 4.25 - Деталі текстури

У меню, яке з'явиться, буде показано всі текстури у вашому проекті. Виберіть Banana_Texture.

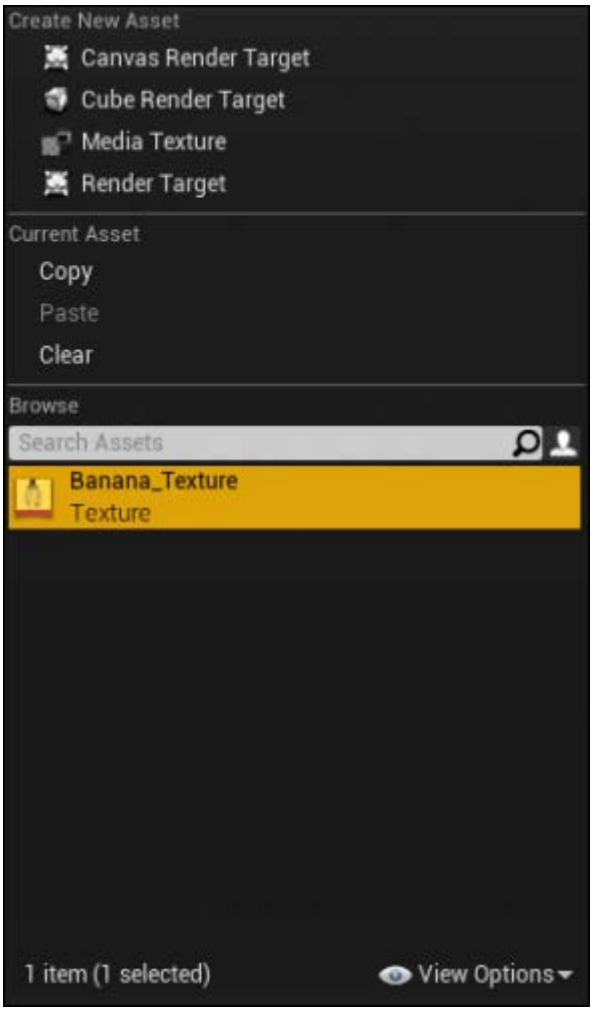

Рис. 4.26 - Нова текстура

Щоб побачити текстуру в сітці попереднього перегляду, потрібно підключити її до вузла результатів. Тримайте лівий клік білого вихідного штифту вузла TextureSample. Перетягніть його до основного вхідного PIN-коду вузла результатів.

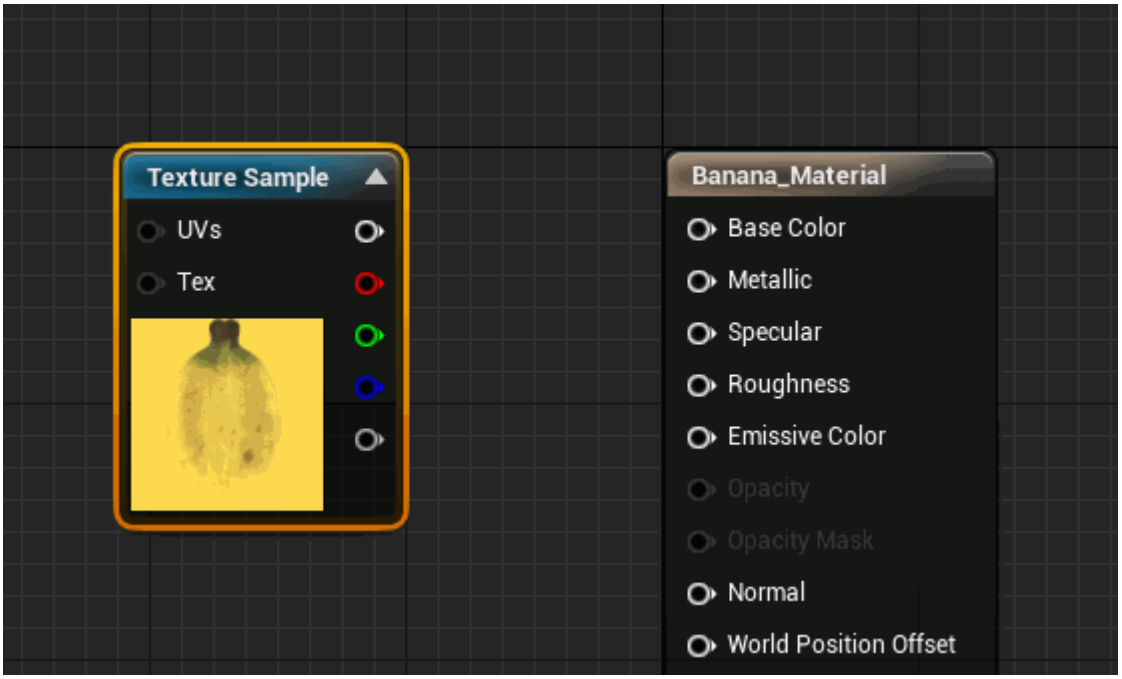

Рис. 4.27- Матеріали та блоки

Поверніться до режиму перегляду, щоб побачити текстуру сітки попереднього перегляду. Поверніть його навколо (утримуючи ліву кнопку миші та перетягнувши), щоб побачити інші деталі.

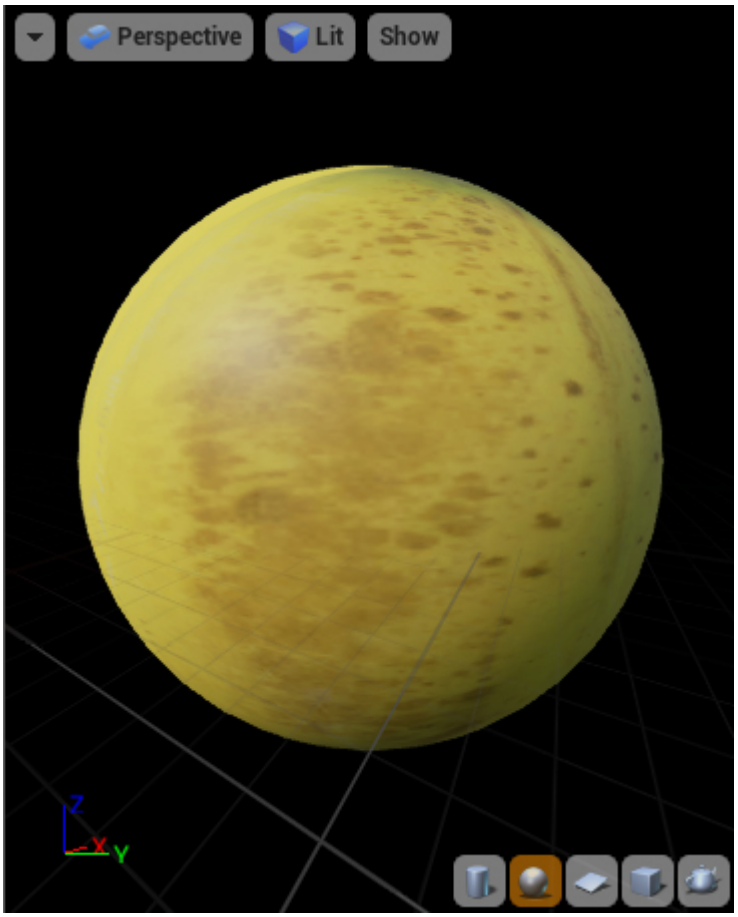

Рис. 4.28 - Вигляд матеріалу

Натисніть кнопку “Застосувати” на панелі інструментів, щоб оновити свій матеріал, і закрийте редактор матеріалів.

4.4.2 Використання матеріалів

Щоб використовувати свій матеріал на банані, потрібно призначити його. Поверніться до браузера контенту та двічі клацніть на Banana_Model. З'явиться наступний редактор:

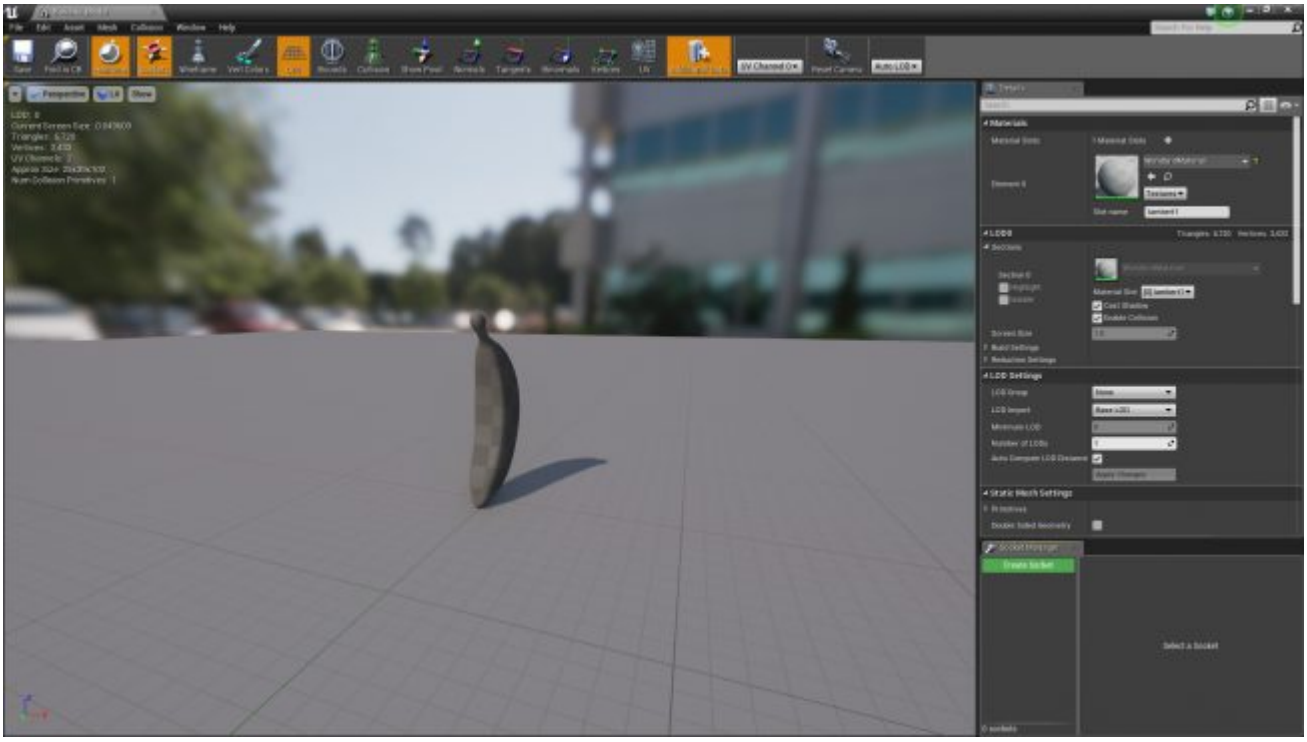

Рис. 4.29 - Об'єкт без матеріалу

Перейдіть на панель “Інформація” та знайдіть розділ “Матеріали”. Натисніть меню, розташоване праворуч від елемента 0, і виберіть Banana_Material.

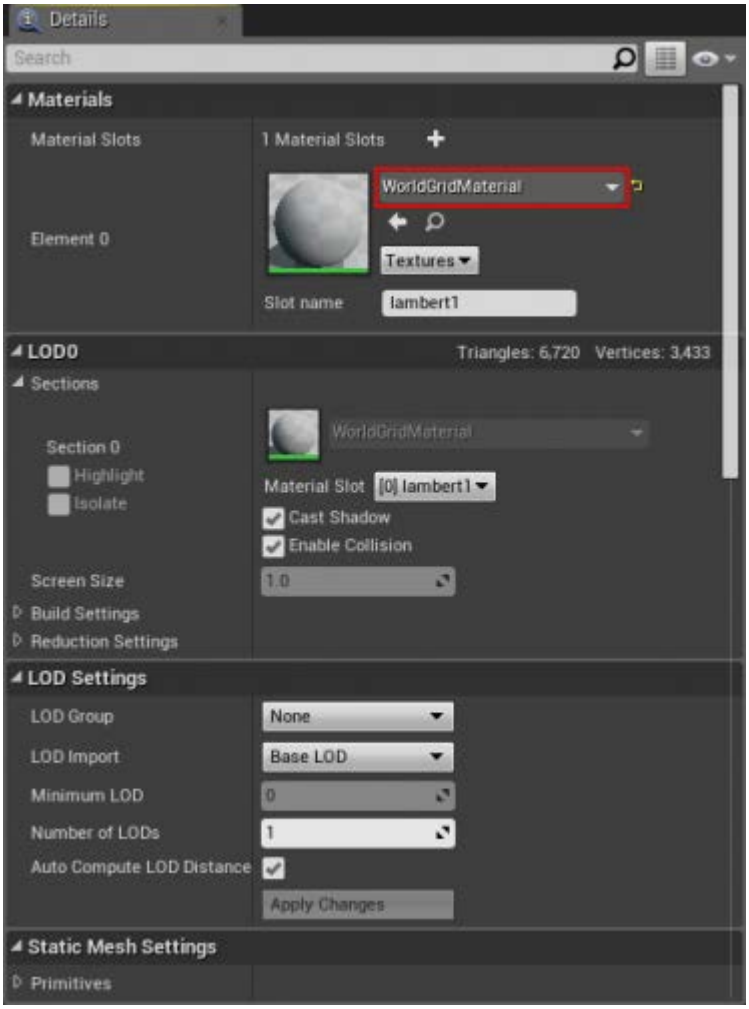

Рис. 4.30 - Поле матеріалу

Закрийте редактор сіток, поверніться до головного редактора та подивіться на Viewport. Ви побачите, що ваш банан тепер має текстуру. Вітаємо, тепер у вас є все, що потрібно, щоб бути дизайнером рівнів!

4.5 Урок №4: Блупрінти

4.5.1 Про блупрінти

Навіть незважаючи на те, що банан і так виглядає дивовижно, він буде виглядати ще краще, обертаючись на рухомому столі.

У найпростішому сенсі, "Бланк" являє собою "річ". Креслення дозволяють створювати власні правила поведінки для ваших об'єктів. Ваш об'єкт може бути чимось фізичним (наприклад, рухомим столом) або чимось абстрактним, наприклад, системою охорони здоров'я.

Хочете зробити рухомий автомобіль? Зробіть схему. А як щодо літаючої свині? Використовуйте креслення.

Як і матеріали, Blueprints використовують систему на основі вузла. Це означає, що все, що вам потрібно зробити, це створювати вузли та пов'язувати їх; Кодування не потрібне!

Примітка. Якщо ви віддаєте перевагу писати код, ви можете використовувати C ++.

Хоча Blueprints прості у використанні, вони не такі швидкі, як C ++. Отже, якщо вам потрібно використовувати щось важке для обчислення, як складний алгоритм, ви повинні розглянути можливість використання C ++.

Навіть якщо ви віддаєте перевагу C ++, існують моменти, коли ідеальним є використання блупрінтів. Ось деякі переваги блупрінтів:

- Як правило, розробляти швидше, використовуючи блупрінти, ніж C ++.

- Легка організація. Ви можете розділити вузли на різні області, такі як функції та графіки.
- Якщо ви працюєте з непрограмістами, модифікація Blueprint проста з огляду на її візуальний та інтуїтивний характер.

Хороший підхід - створити свої об'єкти за допомогою грейбоксів. Якщо вам потрібна додаткова продуктивність, перетягніть їх на C ++.

4.5.2 Створення блупрінту

Перейдіть до браузера вмісту та натисніть “Додати новий”. У списку виберіть “Клас”.

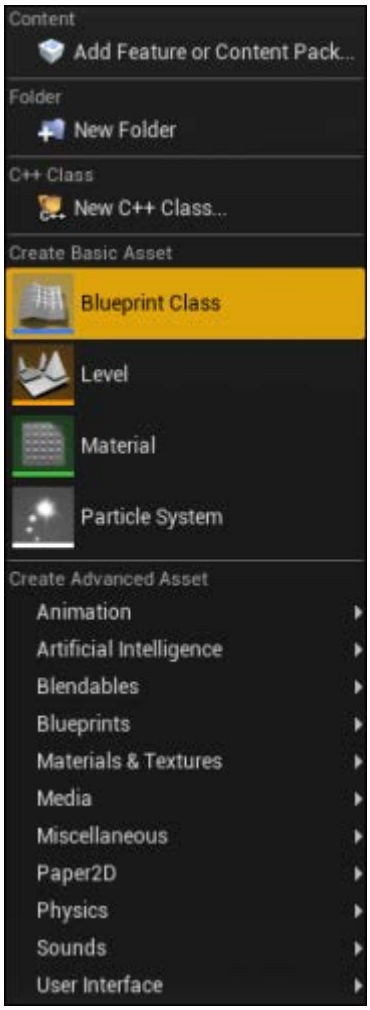

Рис. 4.31 - Блупрінт клас

З'явиться вікно з пропозицією вибрати батьківський клас. Ваш Blueprint успадкує всі змінні, функції та компоненти від обраного вами батьківського класу. Приділіть хвилину, щоб прочитати, що робить кожен клас.

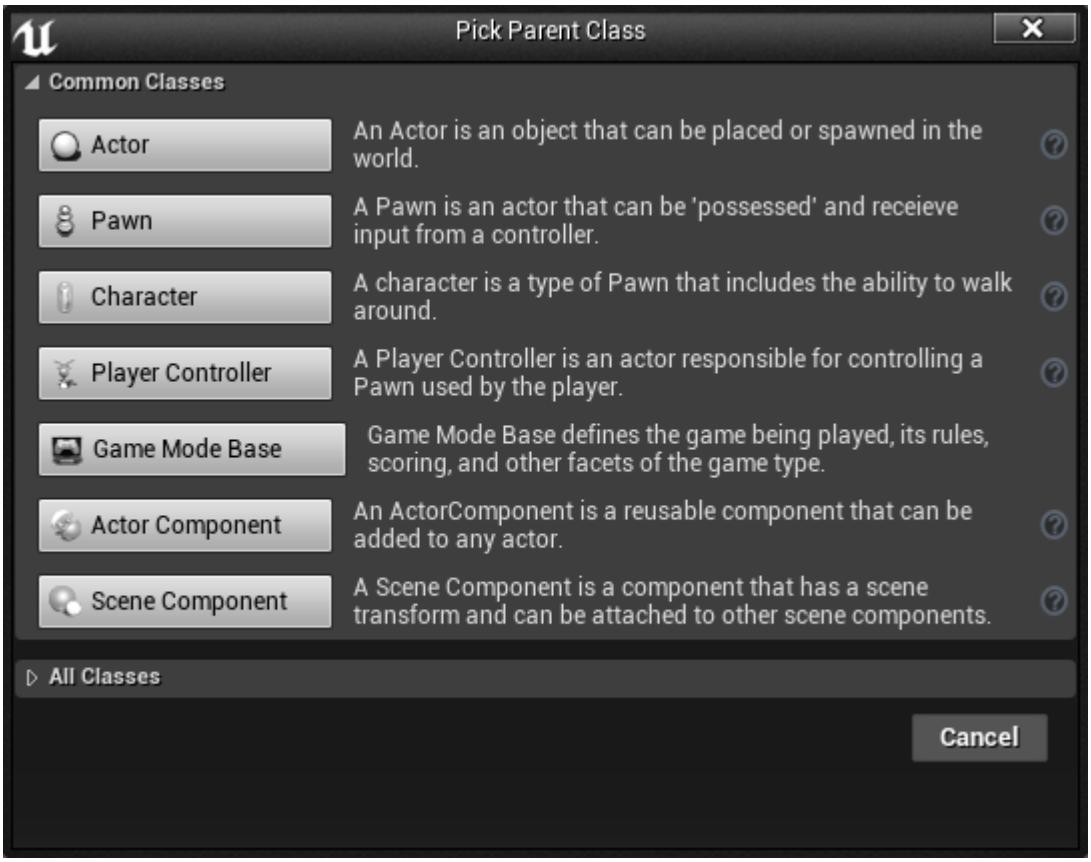

Рис. 4.32 - Вибір перент класу

Оскільки платформа просто перебуватиме на одному місці, клас актора є найбільш підходящим. Виберіть "Актор" і назвіть новий файл Banana_Blueprint.

Рис. 4.33 - Додання блупрінту

Нарешті, двічі натисніть на Banana_Blueprint, щоб відкрити його. Клацніть Open Full Blueprint Editor, якщо з'явиться вікно, подібне до цього:

Рис. 4.34 - Параметру блупрінту

4.5.3 Створення рухомого столу

Для створення рухомого столу вам потрібні дві речі: база та дисплей. Ви можете створити обидва з них за допомогою компонентів.

Що таке компонент?

Якщо Blueprint - це автомобіль, то компоненти є будівельними блоками, що складають машину. Двері, колісні диски та двигун - це всі приклади компонентів.

Проте компоненти не обмежуються лише фізичними об'єктами.

Наприклад, щоб зробити переміщення автомобіля, можна додати компонент руху. Ви навіть можете змусити машину літати, додавши літаючий компонент.

Додавання компонентів

Перш ніж побачити будь-які компоненти, вам потрібно перейти до перегляду Viewport. Натисніть вкладку "Перегляд", щоб перейти на нього. Він виглядає так:

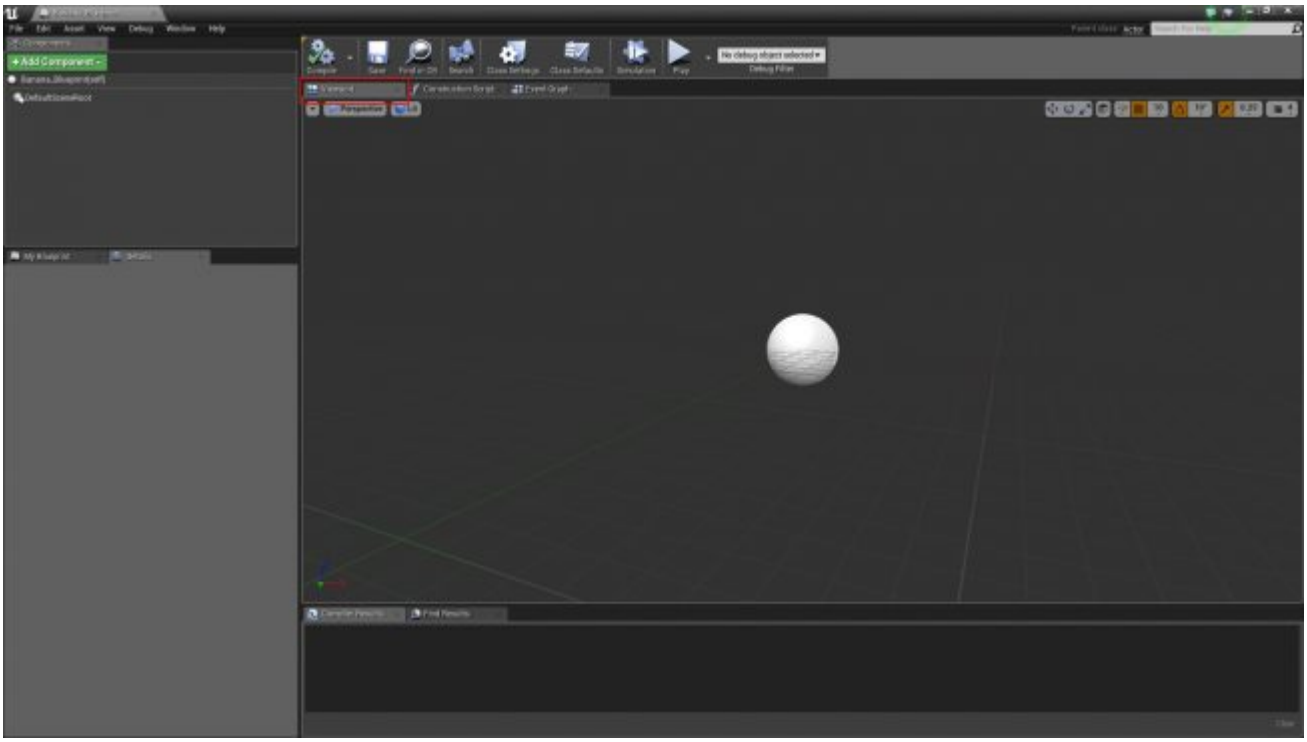

Рис. 4.35 - Перегляд матеріалу

На складі буде використано два компоненти:

- Циліндр: простий білий циліндр. Це буде базою, на якій стоїть банан.
- Статична сітка: цей компонент відобразить бананову сітку.

Щоб додати базу, перейдіть на панель компонентів. Натисніть “Додати компонент” та виберіть “Циліндр”.

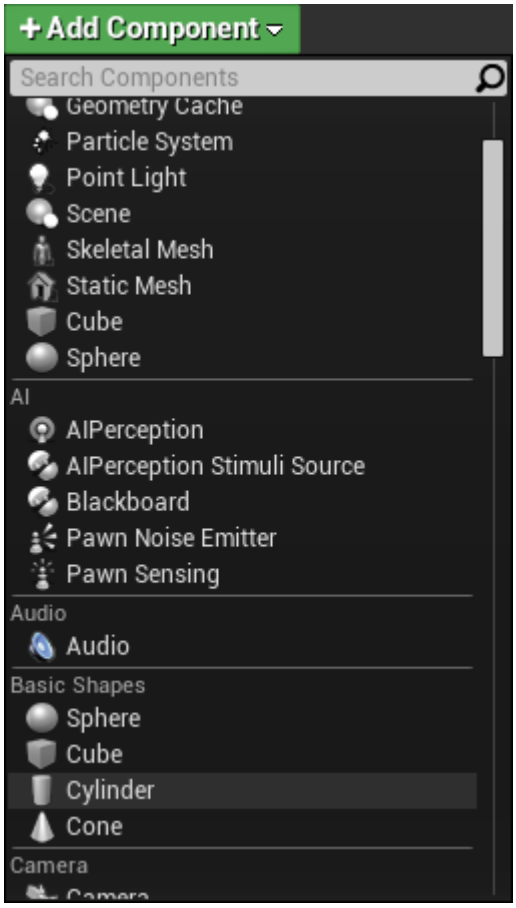

Рис. 4.36 - Додання компоненту

Бажано зробити базу трохи коротшою. Активізуйте маніпулятор масштабу, натиснувши R, а потім можна масштабувати його (точний розмір не має значення, ви можете налаштувати його пізніше, якщо захочете).

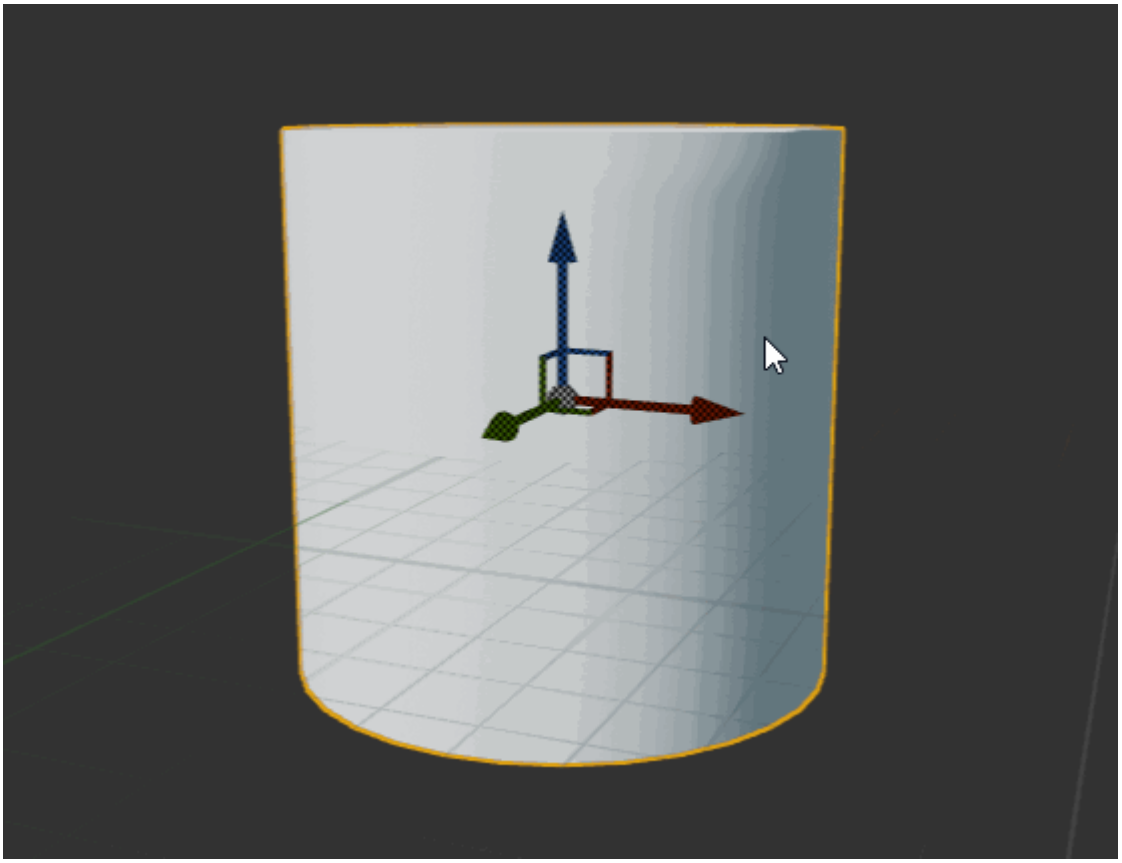

Рис. 4.37 - Маніпулятор переміщення

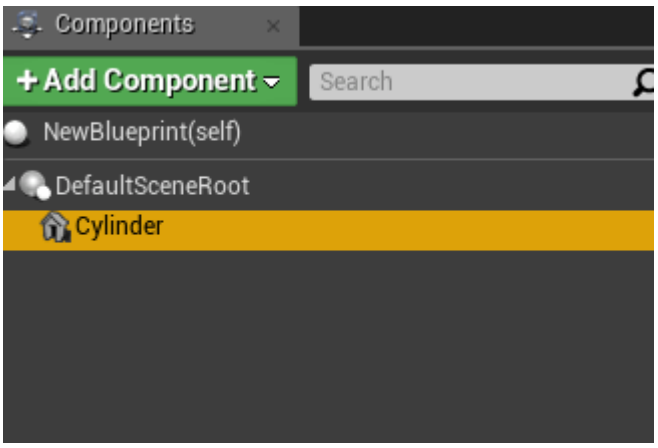

Рис. 4.38 - Додана компонента

Примітка. Якщо ви цього не зробите, наступний компонент буде прикріплений до компоненту циліндра. Це означає, що він також успадкує масштаб компоненту циліндра. Оскільки ви зменшили циліндр, наступний компонент також буде зменшено.

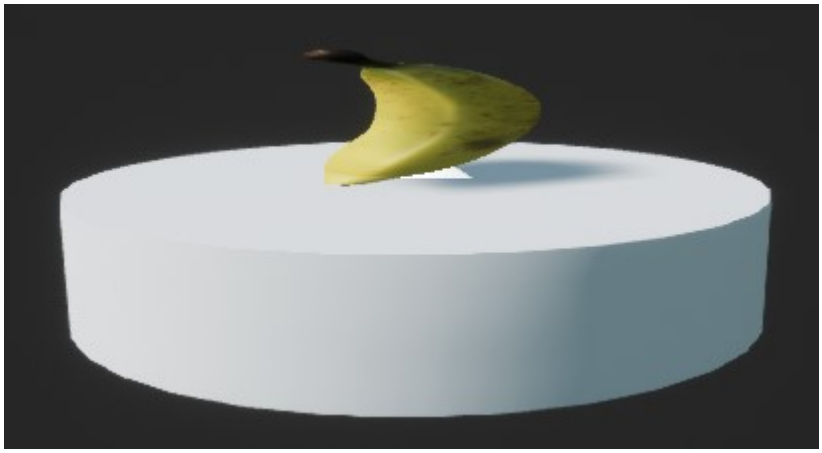

Рис. 4.39 - Результат рівню

Потім натисніть Додати компонент і виберіть зі списку статичну сітку.

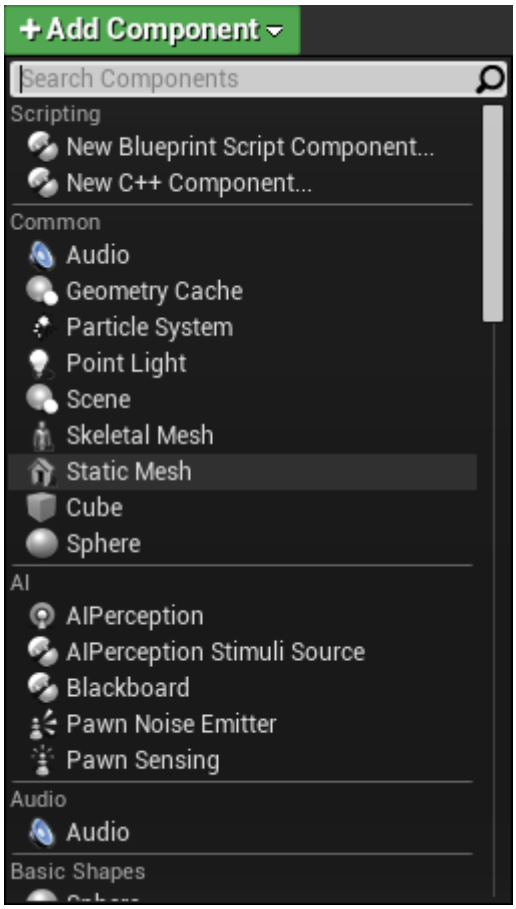

Рис. 4.40 - Додання компоненту

Щоб відобразити банан, виберіть компонент Static Mesh, а потім клацніть вкладку “Деталі”. Клацніть на контекстному меню, розташованому праворуч від Static Mesh, і виберіть Banana_Model.

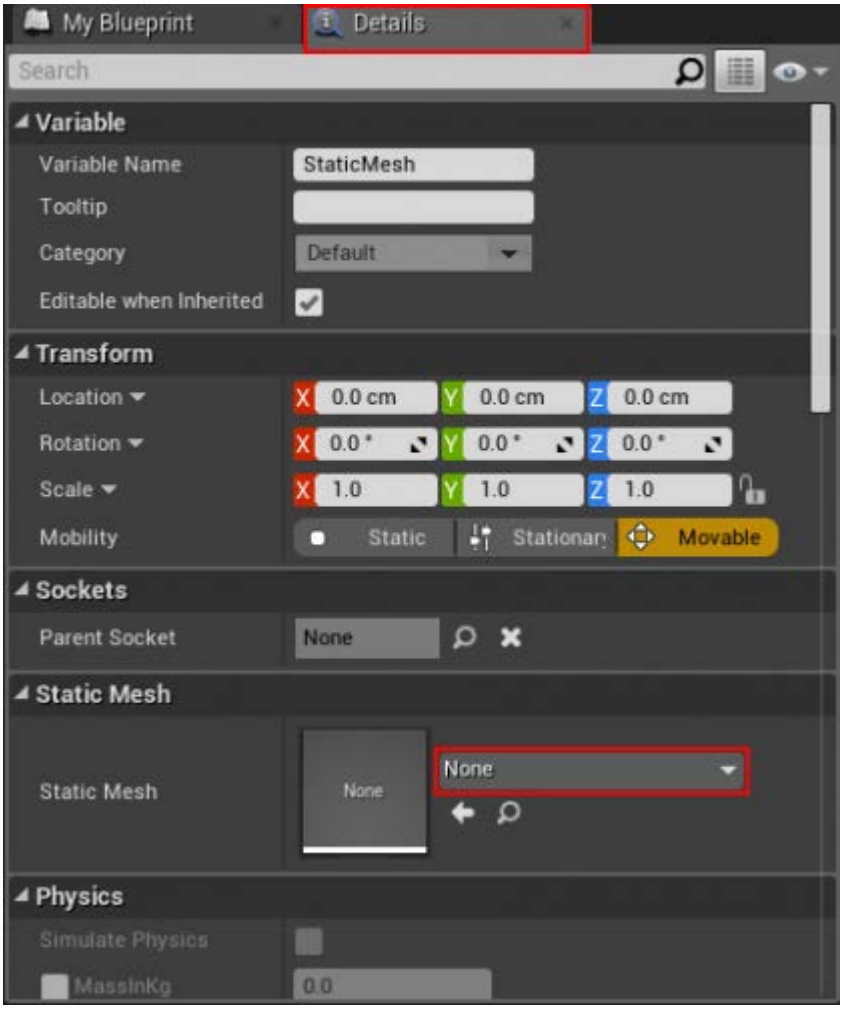

Рис. 4.41 - Параметри мешу

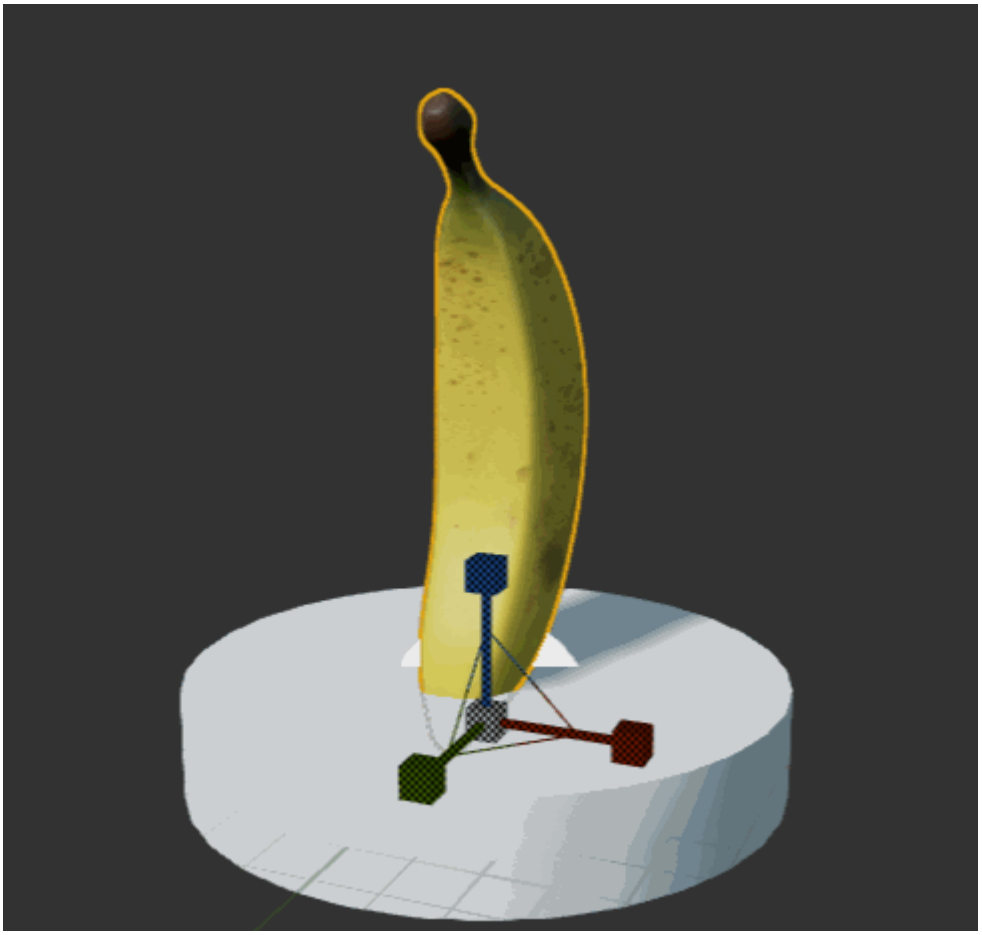

Рис. 4.42 - Маніпулятор масштабування

4.5.4 Про ноди в блупрінтах

На відміну від вузлів у матерілах, вузли у Blueprint мають спеціальні входи під назвою Execution pins. Штифт зліва - це вхід, а штифт справа - вихід. У всіх вузлах буде принаймні один з них.

Якщо вузол має вхідний PIN, він повинен мати з'єднання, перш ніж він може виконуватись. Якщо вузол не підключений, будь-які наступні вузли не виконуватимуться.

Ось приклад:

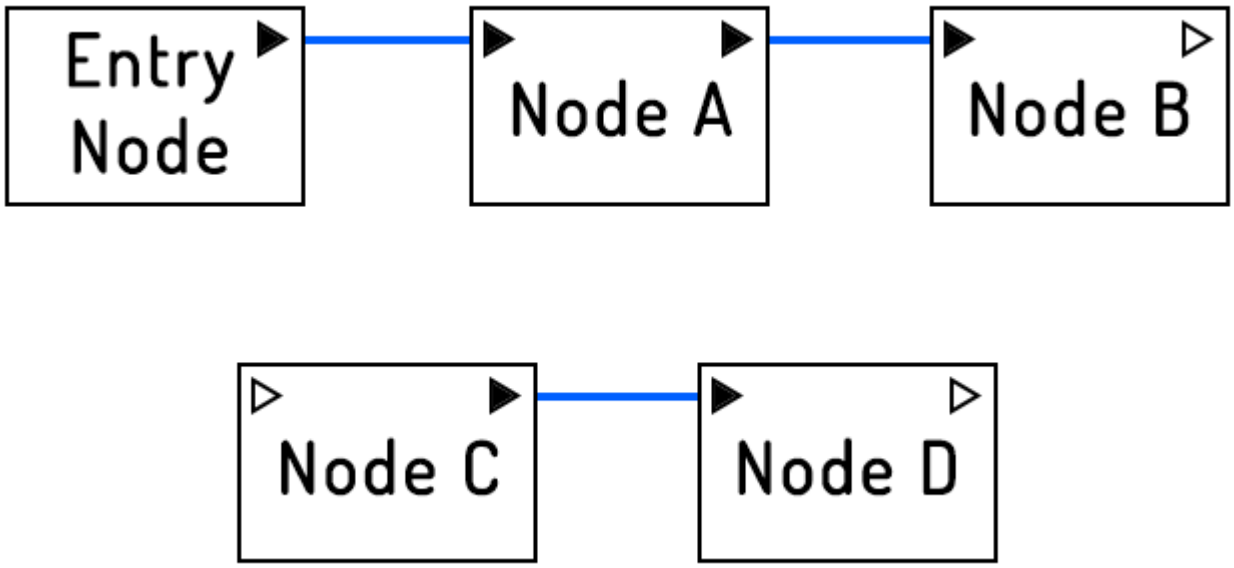

Рис. 4.43 - Логіка нодів

Вузол A і вузол B будуть виконуватись, оскільки їх вхідні контакти мають з'єднання. Вузол C і вузол D ніколи не виконуватимуться, оскільки вхідний контакт Node C не має з'єднання.

4.5.5 Поворот столу

Перш ніж почати, перегляньте панель компонентів. Ви побачите, що Циліндр та Статична сітка не знаходяться на тому самому рівні, що й DefaultSceneRoo. Це тому, що вони прикріплені до DefaultSceneRoot.

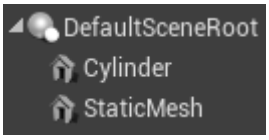

Рис. 4.44 - Компоненти

Якщо ви рухаєте, обертаєте або масштабуєте кореневий компонент, додані компоненти теж будуть виконувати ці операції. Використовуючи цю модель, ви можете обертати циліндр та статичну сітку разом, а не індивідуально.

Створення вузла

Щоб запустити сценарії, поверніться до вкладки “Графік подій”.

Обертати об'єкта настільки просто, що вам потрібно лише створити один вузол. Клацніть правою кнопкою миші на порожній простір на графіку, щоб відкрити меню доступних вузлів. Шукайте AddLocalRotation. Оскільки вам потрібно обертати базу та банан, ви можете просто повернути кореневий компонент. Виберіть AddLocalRotation (DefaultSceneRoot).

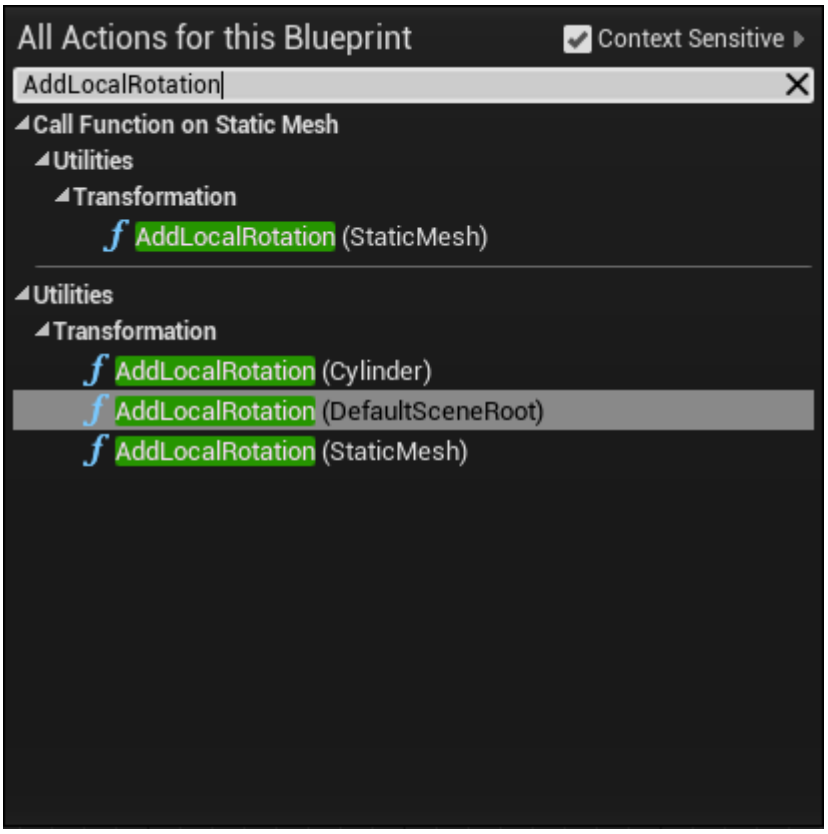

Рис. 4.45 - Додання функції

На вашому графі тепер буде новий вузол AddLocalRotation. Вхід Target автоматично матиме з'єднання з вибраним вами компонентом.

Щоб встановити значення обертання, перейдіть до входу з інтеграцією Delta Rotation і змініть значення Z до 1,0. Це призведе до того, що Blueprint буде обертатися навколо своєї осі Z. Більш високі значення призведуть до швидшого обертання поворотного столу.

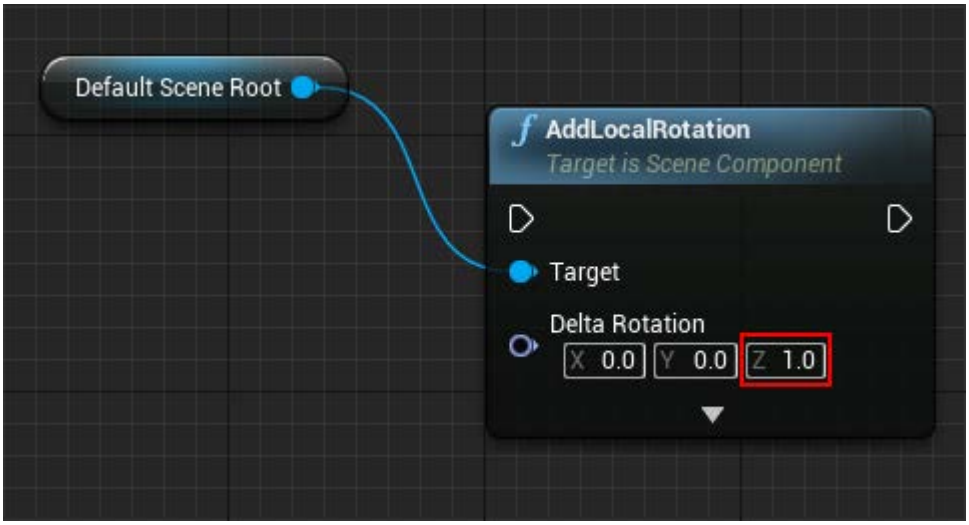

Рис. 4.46 - Підключення блоку

Щоб постійно повертати рухомий столик, вам потрібно викликати AddLocalRotation для кожного кадру. Для виконання вузла для кожного кадру використовуйте вузол Tick Event. Він вже на вашому графіку. Якщо ні, ви можете створити його за допомогою того самого методу, що і раніше.

Перетягніть вихідний штифт вузла Tick Event Tick на вхідний штифт вузла AddLocalRotation.

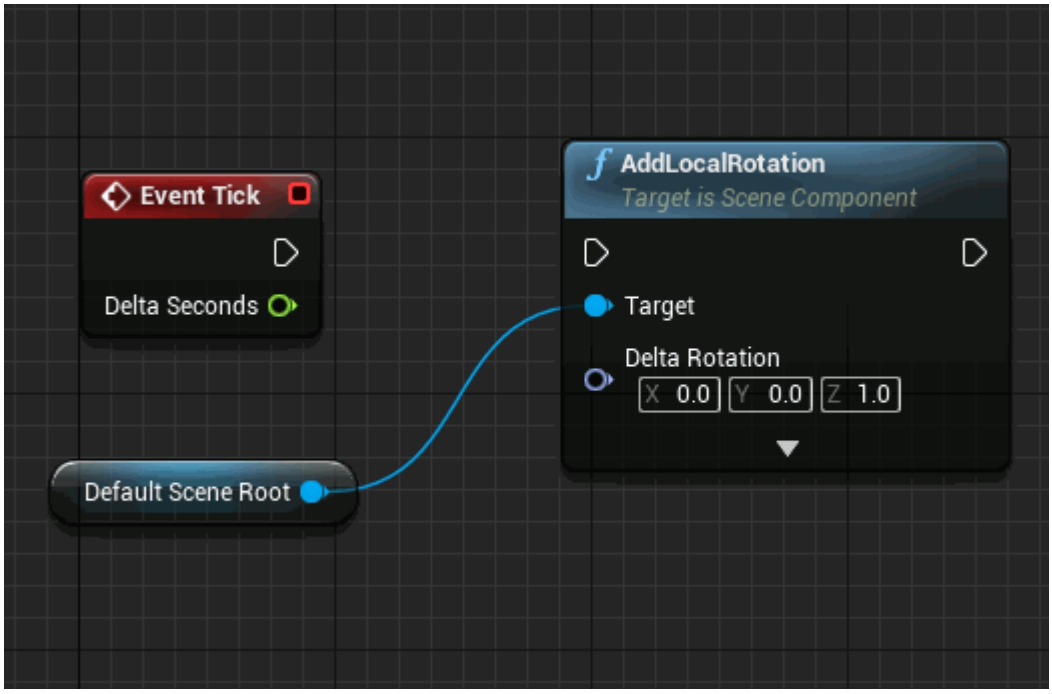

Рис. 4.47 - Додання зв'язку

Примітка. У цій реалізації швидкість обертання залежить від частоти кадрів. Це означає, що рухомий стіл повертатиметься повільніше на більш повільних машинах і навпаки.

Нарешті, перейдіть на панель інструментів і натисніть “Компіляція”, щоб оновити свій проект, а потім закрийте редактор Blueprint.

Рис. 4.48 - Компіляція блупрінту

4.5.6 Додавання блупрінтів на рівень

Перш ніж додавати Blueprint, поверніться до Viewport у головному редакторі та видаліть модель banana. Для цього виберіть модель, а потім виберіть Редагувати \ Видалити або натисніть клавішу “Видалити”.

Додавання схеми - це той самий процес, що й додавання сітки. Тримайте лівий клік на файлі та перетягніть його в Viewport.

Перейдіть до “Панелі інструментів” та натисніть “Грати”, щоб побачити всю вашу важку роботу в дії!

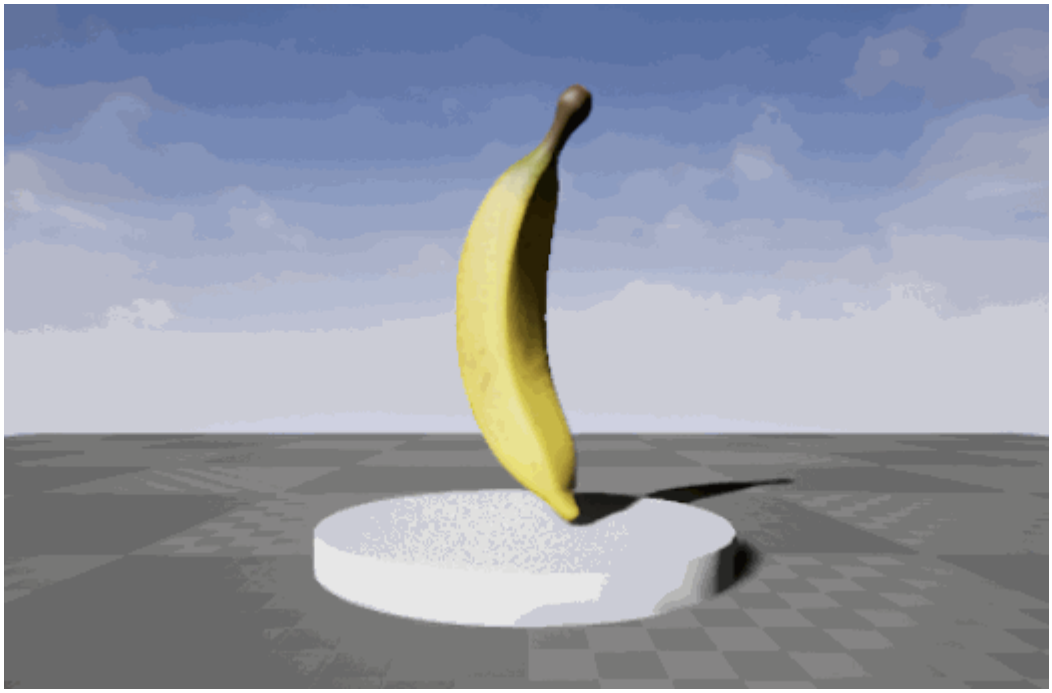

Рис. 4.49 - Результат роботи

4.6 Урок №5: Створення GameMode

Перейдемо до створення GameMode. GameMode визначає саму гру, її правила та умови виграшу. Також встановлює класи за замовчуванням, що використовуються для деяких основних типів геймплею, включаючи Leang, PlayerController та HUD. Перш ніж ми створимо персонажа нашого FPS, нам потрібно створити GameMode, який буде посилатися на нього.

По-перше, ми будемо використовувати майстер створення класів C ++, щоб додати новий клас до нашого проекту.

1. У меню “Файл” виберіть “Додати код до проекту”.

2. Прокрутіть вниз і виберіть GameMode як батьківський клас. Натисніть кнопку “Далі”.

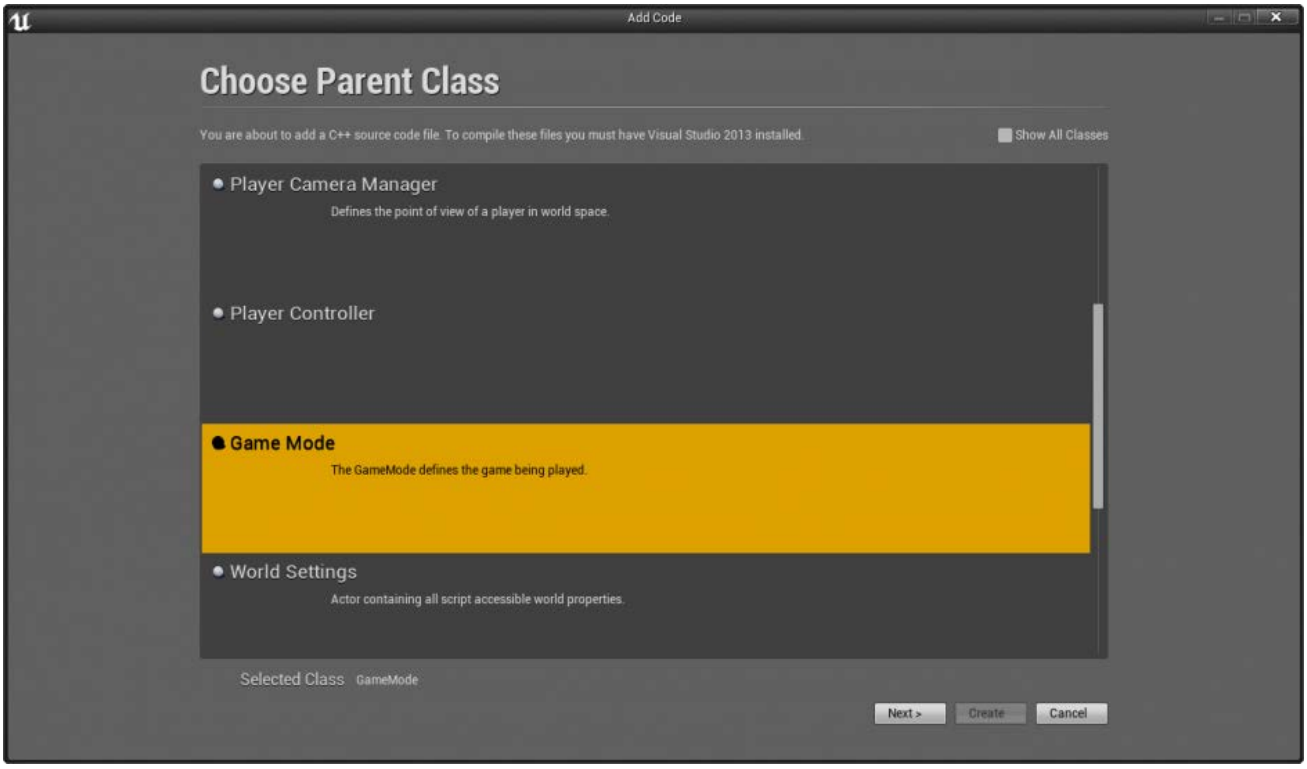

Рис. 4.50 - Вибір перент класу

3. Назвіть новий клас FPSGameMode, а потім натисніть кнопку “Створити”.

4. Натисніть “Так”, щоб відкрити клас у Visual Studio або XCode для редагування.

Оскільки це перший код, який ми додали до проекту, майстер також створить початкові файли, необхідні для компіляції та запуску нашого проекту.

Ми додамо лог-повідомлення до FPSGameMode, так що коли ми почнемо грати на нашому рівні, можна буде побачити, що ми насправді використовуємо наш новий GameMode. Ми додамо код до конструктора FPSGameMode, щоб він запускався, коли починається геймплей.

1. Відкриється код IDE.

2. У Провіднику Solution розгорніть FPSProject> Source> FPSProject.

3. Тут ви побачите файл заголовка для вашого нового класу FPSGameMode, FPSGameMode.h. Двічі клацніть по ньому, щоб відкрити його для редагування.

4. Знайдіть декларацію класу, яка виглядає так:

```
UCLASS()
class FPSPROJECT_API AFPSGameMode : public AGameMode
{
GENERATED_BODY()
};
```

5. Під заголовком GENERATED_BODY () додайте наступні рядки, а потім збережіть файл.

```
virtual void StartPlay() override;
```

Ця декларація функції дозволить вам перевизначити функцію StartPlay (), успадковану від класу `AActor`, щоб ви могли надрукувати повідомлення на екрані, коли починається геймплей.

6. Відкрийте FPSGameMode.cpp. Він знаходиться також у FPSProject> Source> FPSProject.

Якщо ваш проект має інше ім'я, ваш код буде розташовано в [ProjectName]> Source> [ModuleName]. Ім'я модуля відповідає назві вашого проекту, оскільки ви використовували майстер класів C ++.

По-перше, у верхній частині файлу необхідно додати наступний рядок під останнім рядком #include:

```
#include "Engine.h" //for version 4.4+
```

7. Знайдіть конструктор FPSGameMode. Він виглядає як:

```
AFPSGameMode::AFPSGameMode(const FObjectInitializer&
ObjectInitializer)
    : Super(ObjectInitializer)
```

```
{

}
```

Префіксоване ім'я FPSGameMode - це AFPSGameMode, оскільки він походить від класу, який в свою чергу походить від класу Actor.

8. Після конструктора додайте наступні рядки, а потім збережіть файл.

```
// Note that engine version 4.3 changed the method's name to
StartPlay(), because of this engine versions before 4.3, or older
tutorials, use BeginPlay()
void AFPSGameMode::StartPlay()
{
      Super::StartPlay();
      StartMatch();
      if (GEngine)
      {
            GEngine->AddOnScreenDebugMessage(-1, 5.f, FColor::Yellow,
TEXT("HELLO WORLD"));
      }
}
```

Ця функція буде друкувати "HELLO WORLD" на екрані жовтим кольором, коли починається геймплей.

Тепер складемо наш проект, щоб бачити зміни коду, відображені в грі. Якщо ви використовуєте Unreal 4,6 або нижче, вам потрібно закрити редактор, скомпілювати, а потім знову відкрити проект у редакторі для перезавантаження ігрового модуля.

1. Якщо ви використовуєте Visual Studio, переконайтеся, що ви налаштували Visual Studio для компіляції з Unreal Engine.

2. Закрийте Unreal Editor. (Не потрібно в 4.7)

3. Скомпонуйте.

4. Після закінчення збірки відкрийте Unreal Editor, а потім відкрийте FPSProject.

Нам потрібно встановити, щоб проект використовував FPSGameMode як GameMode за замовчуванням.

1. У меню “Редагування” клацніть “Налаштування проекту”.

2. У розділі "Гра" в лівій частині вкладки Налаштування проекту натисніть "Карти та режими".

3. Виберіть FPSGameMode у контекстному меню GameMode Default.

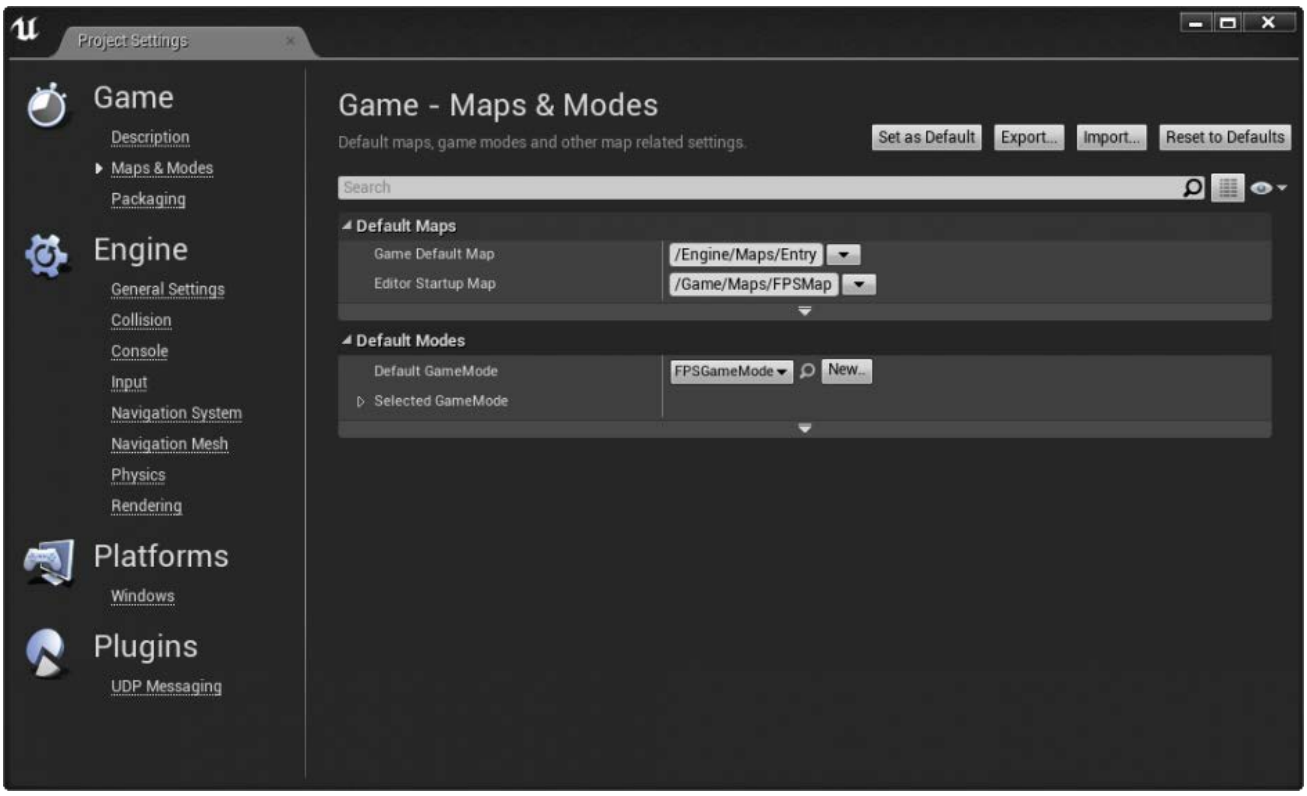

Рис. 4.51 - Додання Game Mode

4. Закрийте меню Налаштування проекту.

5. Натисніть кнопку “Відтворити” на панелі інструментів редактора рівнів. "HELLO WORLD" має відображатися у верхньому лівому куті вікна перегляду.

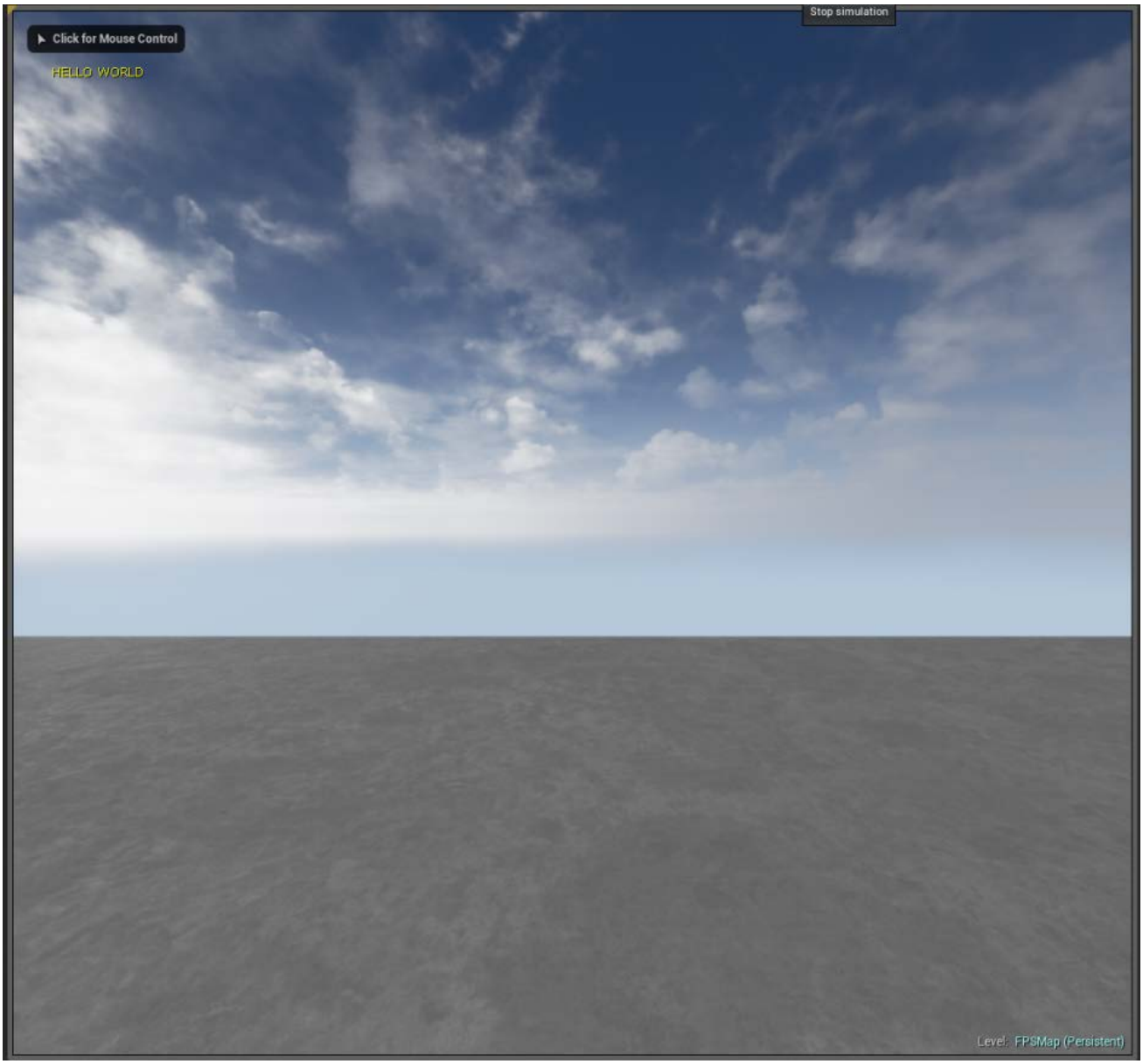

Рис. 4.52 - Scene Outliner

Ви також можете подивитися на Scene Outliner під час вашої гри, щоб побачити FPS GameMode у списку.

6. Натисніть клавішу Escape, щоб вийти з режиму Play in Editor (PIE).

4.7 Урок №6: Створення персонажу

Двигун має вбудований клас під назвою DefaultPawn. Ми хочемо, щоб людина, як аватар, ходила по землі, так що давайте створимо свою власну піхоту для управління. Двигун включає в себе клас, який називається Character, який походить від Pawn, але має вбудовану функціональність для рухів, таких як ходьба, біг та стрибки. Ми будемо використовувати Character як базовий клас для нашої FP-персонажів.

Знову ж таки, ми будемо використовувати майстер класів C ++, щоб додати цей новий клас до нашого проекту. Можна вручну додати файли * .h та * .cpp до свого солюшену Visual Studio для додавання нових класів, але майстер класу C ++ заповнює шаблони заголовків і джерел, які налаштовують для нас спеціальні макроси Unreal, що спрощує процес.

1. У меню “Файл” виберіть “Додати код до проекту”.

2. Прокрутіть вниз і виберіть символ як батьківський клас. Натисніть кнопку Далі.

3. Назвіть новий клас FPSCharacter, а потім натисніть кнопку Створити.

4. Натисніть Так, щоб відкрити клас у Visual Studio для редагування.

5. Visual Studio запитає вас про перезавантаження проекту, оскільки майстер класів C ++ змінив його. Виберіть Перезавантажити.

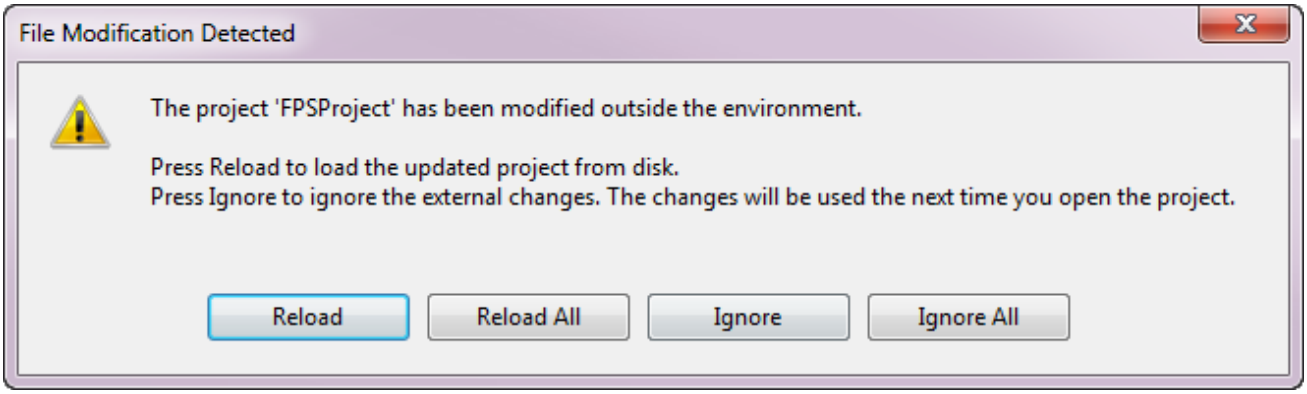

Рис. 4.53 - Інформація про зміну проекту

По-перше, ми будемо редагувати наш GameMode, так що FPSCharacter - це піктограма за замовчуванням, використовувана при запуску геймплея.

1. Спочатку перейдіть на FPSGameMode.h. Нижче GENERATED_BODY () ми тепер додамо конструктор для класу. Тепер клас виглядає так:

```
UCLASS()
class FPSPROJECT_API AFPSGameMode : public AGameMode
{
	GENERATED_BODY()

	AFPSGameMode(const FObjectInitializer& ObjectInitializer);  //
Our added constructor

	virtual void StartPlay() override;
};
```

2. Повернутися до FPS GameMode.c ++. По-перше, потрібно у верхній частині файлу додати наступний рядок під останнім рядком #include:

```
#include "FPSCharacter.h"
```

3. Потім ми знайдемо конструктор, який ми раніше додавали в FPSGameMode.cpp:

```
AFPSGameMode::AFPSGameMode(const class FObjectInitializer&
ObjectInitializer)
	: Super(ObjectInitializer)
{
}
```

4. І після цього до конструктора додаємо наступний рядок:

```
DefaultPawnClass = AFPSCharacter::StaticClass();
```

Це повідомляє GameMode, який виникає для гравця при запуску гри.

Конструктор для FPSGameMode тепер виглядатиме так:

```
AFPSGameMode::AFPSGameMode(const class FObjectInitializer&
ObjectInitializer)
      : Super(ObjectInitializer)
{
      DefaultPawnClass = AFPSCharacter::StaticClass();
}
```

Тепер збережіть файли.

Давайте також додамо на екрані повідомлення до FPSCharacter, щоб ми могли бути впевнені, що наш новий клас правильно використовується.

1. Відкрийте FPSCharacter.h. Він розташований у FPSProject> Source> FPSProject.

Якщо ваш проект має інше ім'я, ваш код буде розташовано в [ProjectName]> Source> [ModuleName]. Ім'я модуля відповідає назві вашого проекту, оскільки ви використовували майстер класу C ++.

2. Знайдіть декларацію класу, яка виглядає так:

```
class FPSPROJECT_API AFPSCharacter : public ACharacter
{
      GENERATED_BODY()

};
```

3. У розділі GENERATED_BODY () додайте наступний рядок, а потім збережіть файл.

```
virtual void BeginPlay() override;
```

Ця декларація функції дозволить вам перевизначити функцію BeginPlay (), успадковану від класу AActor, щоб ви могли надрукувати повідомлення на екрані, коли починається геймплей.

4. Запустіть FPSCharacter.cpp та додайте наступні рядки, а потім збережіть файл.

```
void AFPSCharacter::BeginPlay()
{
      Super::BeginPlay();

      if (GEngine)
      {
            GEngine->AddOnScreenDebugMessage(-1, 5.f, FColor::Blue,
TEXT("We are using FPSCharacter!"));
      }
}
```

Це визначення функції буде друкувати "Ми використовуємо FPSCharacter!" на екрані синім текстом, коли починається геймплей.

Ми знову закриємо редактор та складемо проект, щоб ми бачили зміни коду, відображені в грі. (Зверніть увагу, що ви повинні мати можливість перезавантаження, якщо ви використовуєте останню версію двигуна, а це означає, що ви можете компілювати, не закриваючи редактор Unreal.)

1. Закрийте Unreal Editor.

2. Скомпілювати.

3. Після закінчення збірки відкрийте Unreal Editor, а потім відкрийте FPSProject.

4. Натисніть кнопку “Відтворити” на панелі інструментів редактора рівня. Ваш новий персонаж ще не має контролю над рухом, тому ви не зможете пересуватися на рівні. Отже, якщо ви застрягли і не можете переміститися, ви правильно використовуєте FPSCharacter! Ваш журнал також повинен відображатися на екрані.

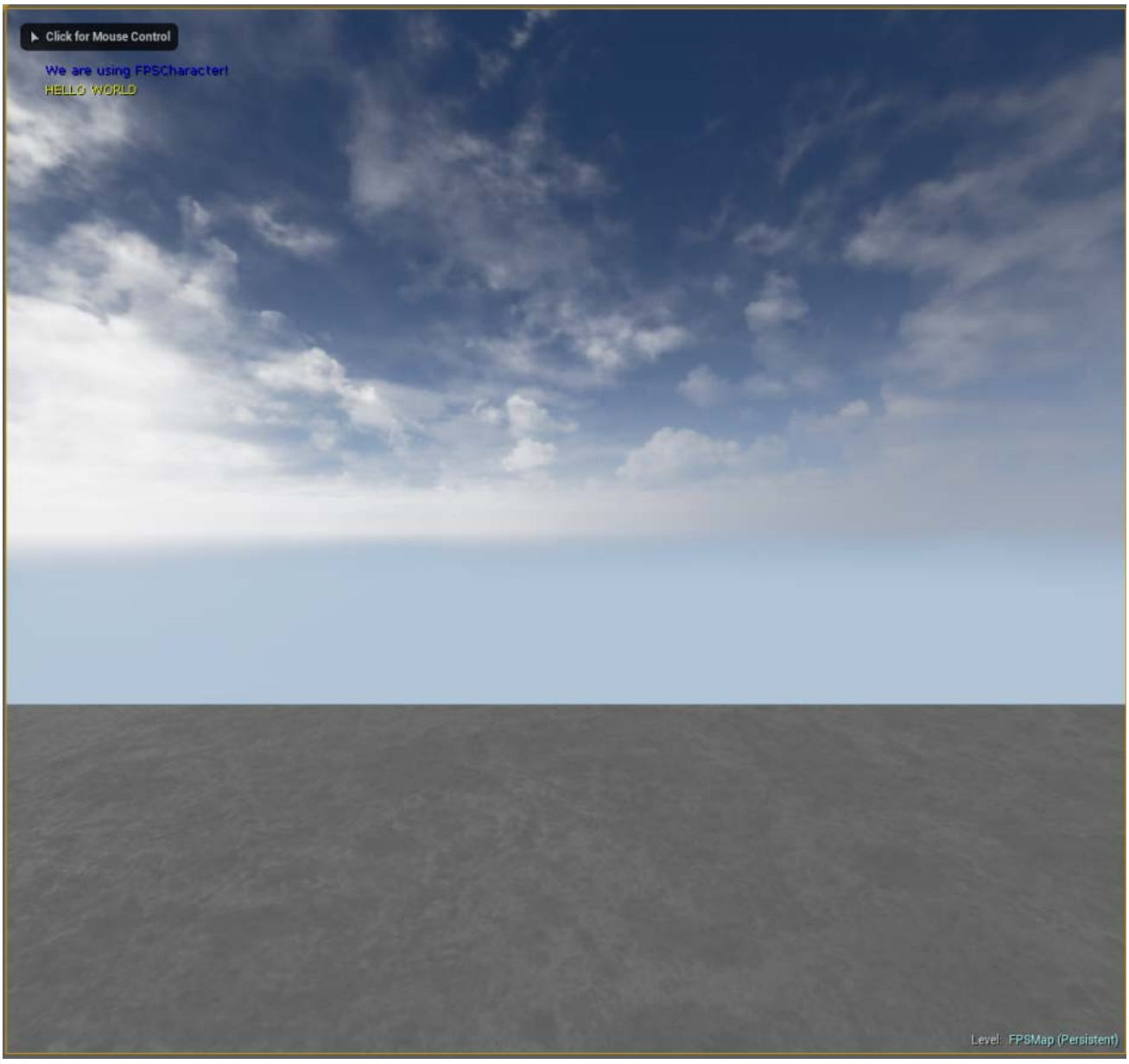

Рис. 4.54 - Результат роботи рівня

Ви також можете подивитися в Scene Outliner під час гри, щоб переглянути список персонажів FPS.

5. Натисніть клавішу Escape, щоб вийти з режиму Play in Editor (PIE).

4.8 Урок №8: Контроль камери

Додамо можливість подивитися навколо та керувати мишею.

Спочатку давайте встановимо відображення осі для Turn і LookUp.

1. У меню “Редагування” клацніть Налаштування проекту.

2. У розділі "Двигун" в лівій частині вкладки Налаштування проекту натисніть на Вхід.

3. У розділі "Прив'язки" натисніть значок "плюс" поруч з Осінними картами.

4. Введіть "Обернути" у відображуване текстове поле, а потім натисніть стрілку ліворуч від текстового поля, щоб розгорнути параметри прив'язки до осі.

5. У спадному меню виберіть MouseX.

6. У розділі "Прив'язки" натисніть значок "плюс" поруч із "Осінними картами".

7. Введіть "LookUp" у відображуване текстове поле, а потім натисніть стрілку ліворуч від текстового поля, щоб розгорнути параметри прив'язки до осі.

8. У спадному меню виберіть MouseY і введіть "-1" для шкали. Ваші параметри введення мають виглядати наступним чином:

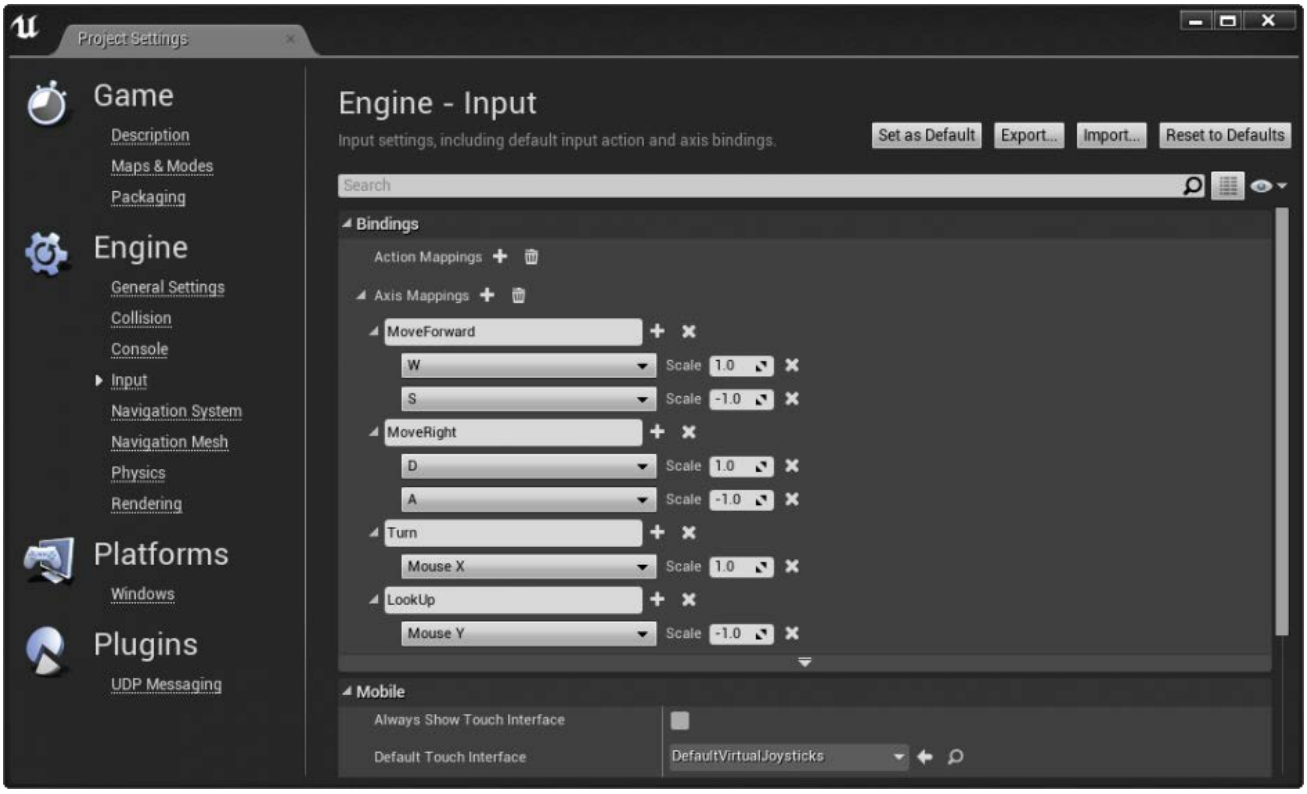

Рис. 4.55 - Налаштування інпуту

9. Закрийте меню Налаштування проекту.

Тепер давайте додамо деякий код для обробки цих входів.

Клас символів визначає нам дві необхідні функції: AddYawInput і AddPitchInput.

Якщо ми хочемо виконати додаткову обробку, наприклад, додавання підтримки для чутливості чи інверсії осі, ми можемо надати власні функції, щоб відрегулювати значення, перш ніж передавати їх до цих функцій, але в цьому випадку зв'яжімо наші дані безпосередньо з ними.

1. Додайте наступні рядки до SetupPlayerInputComponent у FPSCharacter.cpp:

```
   InputComponent->BindAxis("Turn", this,
&AFPSCharacter::AddControllerYawInput);
   InputComponent->BindAxis("LookUp", this,
&AFPSCharacter::AddControllerPitchInput);
```

Оскільки ми змінили існуючу функцію, а не створювали нову функцію, ми можемо скомпілювати його в редакторі.

1. Поверніться до редактора Unreal і натисніть кнопку “Компіляція”.

2. У нижньому правому кутку екрана з'явиться повідомлення про компіляцію коду C ++. Зачекайте, поки він закінчиться - він скаже компіляцію завершеним, а потім зникне.

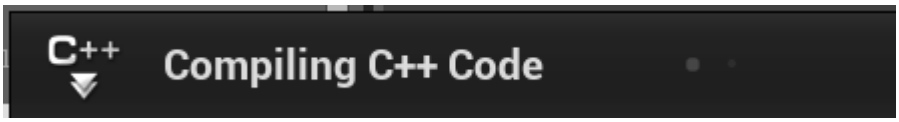

Рис. 4.56 - Компіляція коду

3. Натисніть кнопку "Відтворити" на панелі інструментів редактора рівнів. Тепер ви можете керувати напрямком камери за допомогою миші.

4. Натисніть "Escape", щоб вийти з режиму Play in Editor (PIE).

4.9 Урок №9: Стрибки

Тепер давайте додамо стрибки на нашу здатність до руху. Наші рухи та кроки управління камерою використовували осі відображення, які обробляють безперервні входи, необхідні для цих типів елементів керування. Існують також відображення дій, які стосуються вхідних даних для дискретних подій.

ActionMappings

Карта дискретної кнопки або клавіші натискає на "дружнє ім'я", яке пізніше буде пов'язане з поведінкою на основі подій. Кінцевим ефектом є те, що натискання (і / або випуск) клавіші, кнопки миші або кнопки клавіатури безпосередньо викликає певну поведінку гри.

Додамо нове відображення дії, яке називається Jump в редакторі.

1. У меню “Редагування” клацніть Налаштування проекту.

2. У розділі "Двигун" в лівій частині вкладки Налаштування проекту натисніть на Вхід.

3. У розділі "Прив'язки" натисніть значок "плюс" поруч із розділенням дій.

4. Клацніть на стрілці ліворуч від елементів дій, щоб розгорнути параметри мапування дій.

5. Введіть "Перейти" у текстовому полі.

6. Розгорніть спадне меню і виберіть Пробіл.

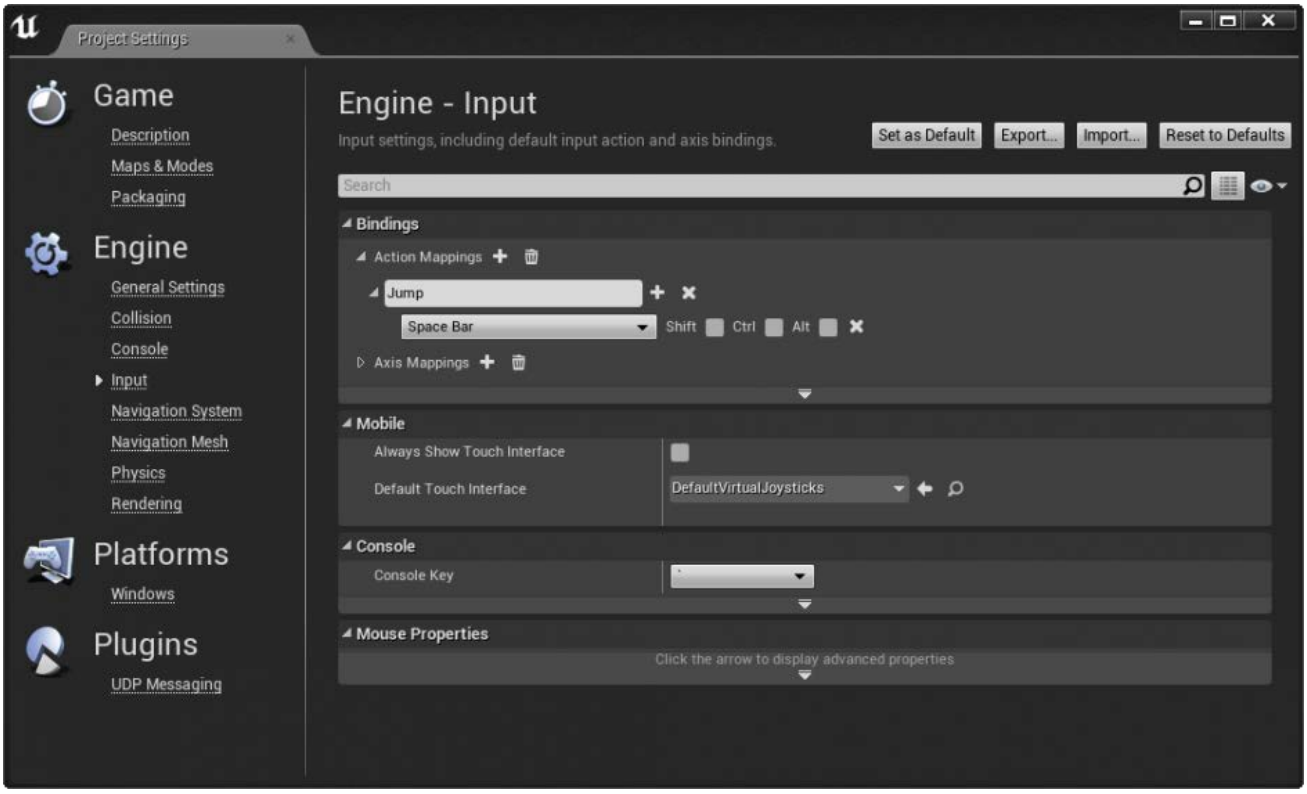

Рис. 4.57 - Налаштування інпуту

7. Закрийте меню Налаштування проекту.

Тепер ми хочемо пов'язувати цю дію з деяким кодом, який призведе нашого персонажа до стрибка.

Якщо ми подивимося на Character.h, ми можемо бачити, що вбудована підтримка стрибка, прив'язана до змінної bPressedJump. Отже, все, що нам потрібно зробити, це встановити прапорець 1, коли натискається дія стрибка, а 0, коли воно вийде. Для цього нам потрібні дві функції.

1. У FPSCharacter.h додайте наступні декларації про загальнодоступні функції:

```
//sets jump flag when key is pressed
UFUNCTION()
void OnStartJump();
//clears jump flag when key is released
UFUNCTION()
void OnStopJump();
```

2. У FPSCharacter.cpp ми можемо реалізувати їх дуже просто:

```
void AFPSCharacter::OnStartJump()
{
    bPressedJump = true;
}
void AFPSCharacter::OnStopJump()
{
    bPressedJump = false;
}
```

Нарешті, нам потрібно пов'язати Action Jump з нашими новими функціями.

3. У SetupPlayerInputComponent додайте наступне:

```
   InputComponent->BindAction("Jump", IE_Pressed, this,
&AFPSCharacter::OnStartJump);
   InputComponent->BindAction("Jump", IE_Released, this,
&AFPSCharacter::OnStopJump);
```

Оскільки ми додали нові функції, а не просто змінювали існуючі функції, як на попередньому кроці, нам потрібно скомпілювати в Visual Studio, а не в редакторі.

1. Закрийте Unreal Editor.

2. Скомпілювати.

3. Після закінчення збірки відкрийте Unreal Editor, а потім відкрийте FPSProject.

4. Натисніть кнопку “Відтворити” на панелі інструментів редактора рівня.

5. Натисніть пробіл, щоб стрибати! Тепер вам слід мати хороший набір початкових елементів керування рухом, рухаючись і стрибаючи WASD, керуючи камерою мишею та стрибаючи.

6. Натисніть клавішу Escape, щоб вийти з режиму Play in Editor (PIE).

4.10 Урок №10: Меш для персонажа

Тепер давайте себе тілом у світі. Клас символів створює нам об'єкт SkeletalMeshComponent за замовчуванням, тому все, що потрібно знати, - це те, що потрібно використовувати для використання SkeletalMesh. Давайте зробимо проект свого класу FPSCharacter, щоб ми могли легко встановити цей актив і маніпулювати будь-якими майбутніми компонентами, які ми хочемо додати. Почнемо з імпорту третьої особи скелетної сітки. Врешті-решт, ми встановимо його таким чином, що є одна сітка, яку гравець бачить, і одна сітка, яку інші гравці бачитимуть у режимі багатокористувацької.

1. Завантажте такий файл zip і розпакуйте його, щоб отримати файл третьої особи.

https://d26ilriwvtzlb.cloudfront.net/1/10/GenericMale.zip

2. Відкрийте Unreal Editor.

3. Клацніть правою кнопкою миші в браузері вмісту та виберіть меню Імпорт до / Гра у меню, що з'явиться.

4. Перейдіть до будь-якого місця, де ви зберегли файл FBX, потім виберіть його та натисніть кнопку Відкрити.

5. Відкрийте розкривне меню Додатково та перевірте імпорт матеріалів, а потім натисніть кнопку Імпортувати.

6. Натисніть на значок "Зберегти" в браузері вмісту, щоб зберегти нову скелетну сітку та пов'язані з нею активи.

Тепер ми можемо створити Синтез нашого класу FPSCharacter і призначити цю нову скелетну сітцю SkeletalMeshComponent.

1. Клацніть правою кнопкою миші в браузері вмісту та виберіть Нова папка. Назвіть цю нову папку Blueprints.

2. Двічі клацніть папку, щоб відкрити її.

3. Клацніть спадне меню New і виберіть Blueprint.

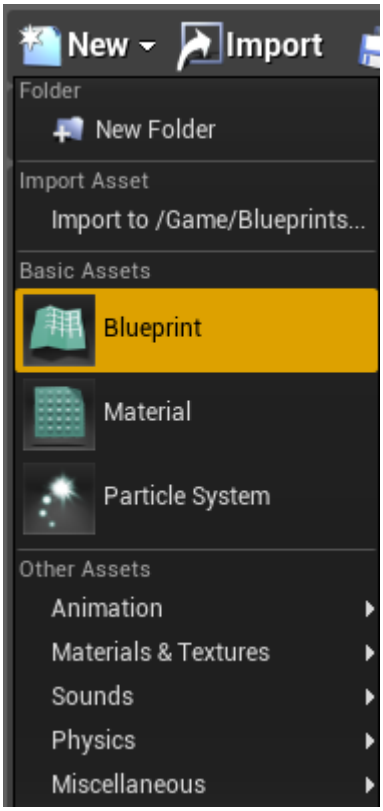

Рис. 4.58 - Додання блупрінту

4. Розкрийте спадне меню Custom Class і введіть "FPSCharacter" у вікно пошуку.

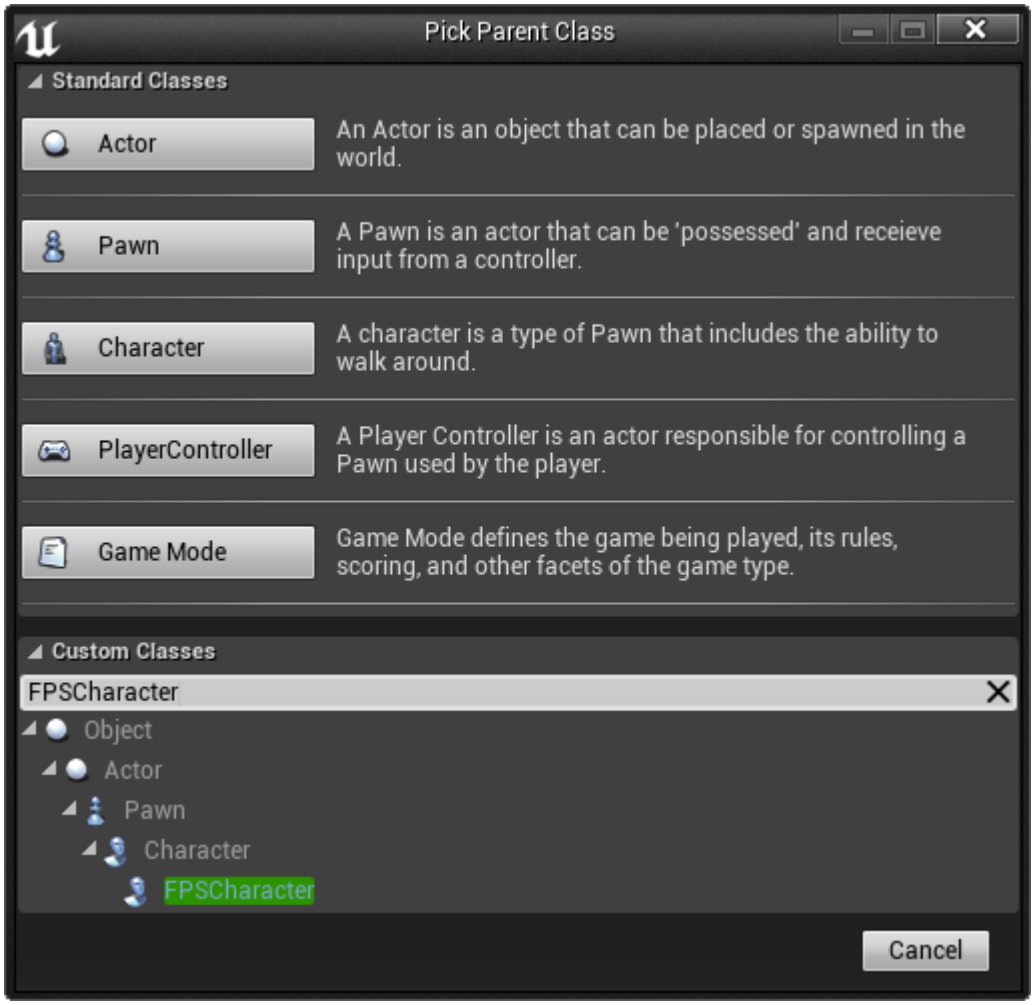

Рис. 4.59 - Вибір перент класу

5. Клацніть на FPSCharacter, щоб вибрати його як батьківський клас для вашого нового Blueprint, а потім натисніть кнопку Вибрати.

6. Назвіть цей новий Blueprint BP_FPSCharacter, потім двічі клацніть його значок, щоб відкрити його.

Редактор Blueprint відкриється у режимі компонентів, щоб ми могли легко встановити сітку третьої особи.

1. Клацніть компонент Mesh на вкладці “Компоненти”.

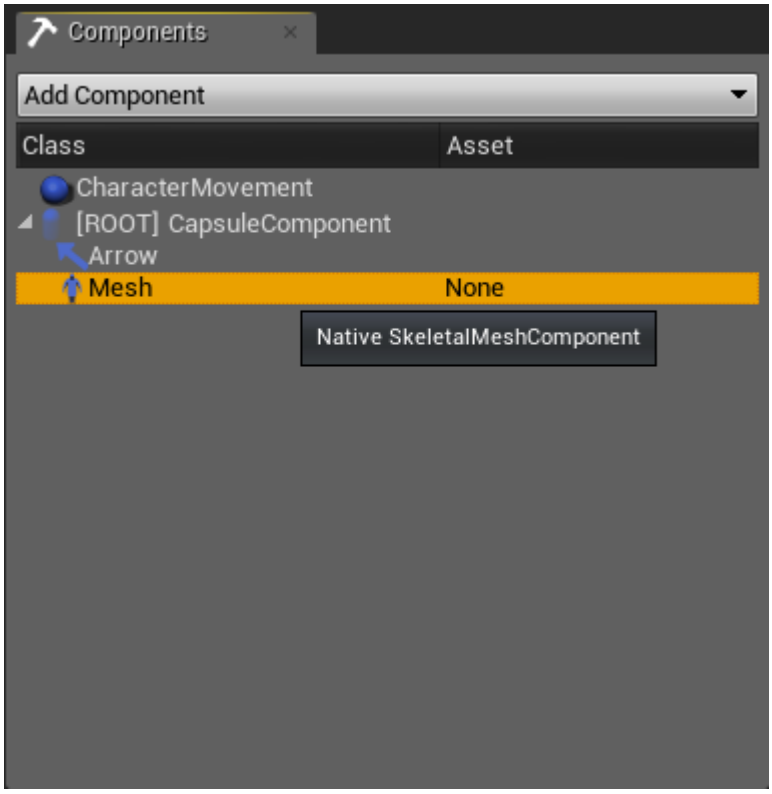

Рис. 4.60 - Додання компоненту

2. На вкладці “Деталі” перейдіть до розділу “Мережа”. Натисніть спадне меню, яке говорить "Ніхто", а потім виберіть нещодавно завантажений актив "Скелетний сітка". Можливо, вам доведеться змінити розмір вкладки “Деталі”, щоб побачити це меню.

3. Вирівняйте SkeletalMeshComponent до CapsuleComponent, встановивши його Z-адресу на -88 на вкладці “Деталі”.

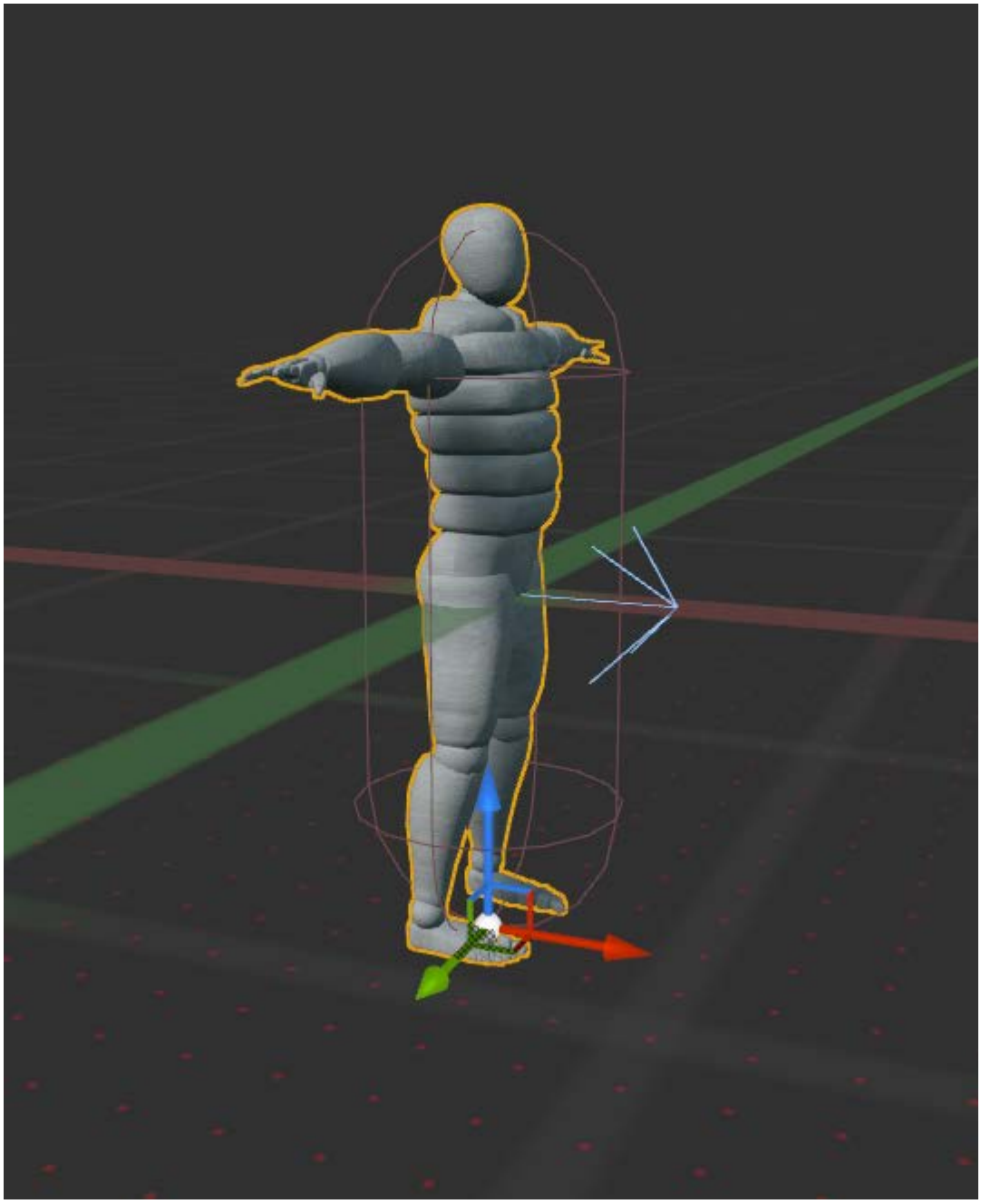

Рис. 4.61 - Результат

Використовуючи власні активи Скелетної сітчини, вам може знадобитися відкоригувати їх по-різному, але загальною метою завжди є наявність сітки, що містяться в CapsuleComponent, і в тому ж напрямку, до якого вказує ArrowComponent. Це забезпечить правильний переміщення вашого персонажа через світ. Ви також можете переміщати компоненти навкруги за допомогою віджетів у вікні перегляду, а не встановлювати значення на вкладці “Деталі”.

4. Скомпонуйте та збережіть свій проект, а потім закрийте редактор креслень.

Тепер нам потрібно сказати нашим GameMode використовувати наш клас Blueprint для пішохода програвача, а не клас FPSCharacter, який ми встановили раніше.

1. Перейдіть до Visual Studio.

2. Перейдіть до конструктора FPSGameMode в FPSGameMode.cpp і замініть існуюче призначення DefaultPawnClass:

```
DefaultPawnClass = AFPSCharacter::StaticClass();
```

з наступним кодом:

```
// set default pawn class to our Blueprinted character
static ConstructorHelpers::FClassFinder<APawn>
PlayerPawnObject(TEXT("Pawn'/Game/Blueprints/BP_FPSCharacter.BP_FPSCha
racter_C'"));
if (PlayerPawnObject.Class != NULL)
{
    DefaultPawnClass = PlayerPawnObject.Class;
}
```

Цей код знайде клас, згенерований вашим проектом, і призначить його як клас закладок за умовчанням. (Зверніть увагу на "_C" суфікс на шляху до ресурсу, це те, що відрізняє фактичний клас, що використовується в грі, з активу Blueprint, який є концепцією лише для редактора). На цьому етапі ви також можете видалити #include "FPSCharacter .h "з вершини FPSGameMode.cpp, оскільки ви вже не звертаєтесь до класу FPSCharacter C ++.

Зверніть увагу: якщо ви помістіть свій проект у іншу папку в дереві ресурсів, ви можете отримати повний шлях, клацнувши його правою кнопкою миші у

браузері вмісту та вибравши “Копіювати довідку”. Повний шлях буде поміщений у ваш буфер обміну для зручного вклеювання.

1. Поверніться до редактора Unreal і натисніть кнопку “Компіляція”.

2. У нижньому правому кутку екрана з'явиться повідомлення про компіляцію коду C ++. Зачекайте, поки він закінчиться - він скаже компіляцію завершеним, а потім зникне.

3. Натисніть кнопку "Відтворити" на панелі інструментів редактора рівнів. Якщо ви перемістите камеру навколо, ви зможете побачити тінь вашого персонажа.

4. Натисніть Shift + F1, щоб відновити курсор миші, потім натисніть кнопку “Витягти” на панелі інструментів. Ви більше не володієте персонажем, тому ви можете вільно переміщати камеру та побачити сітку вашого персонажа.

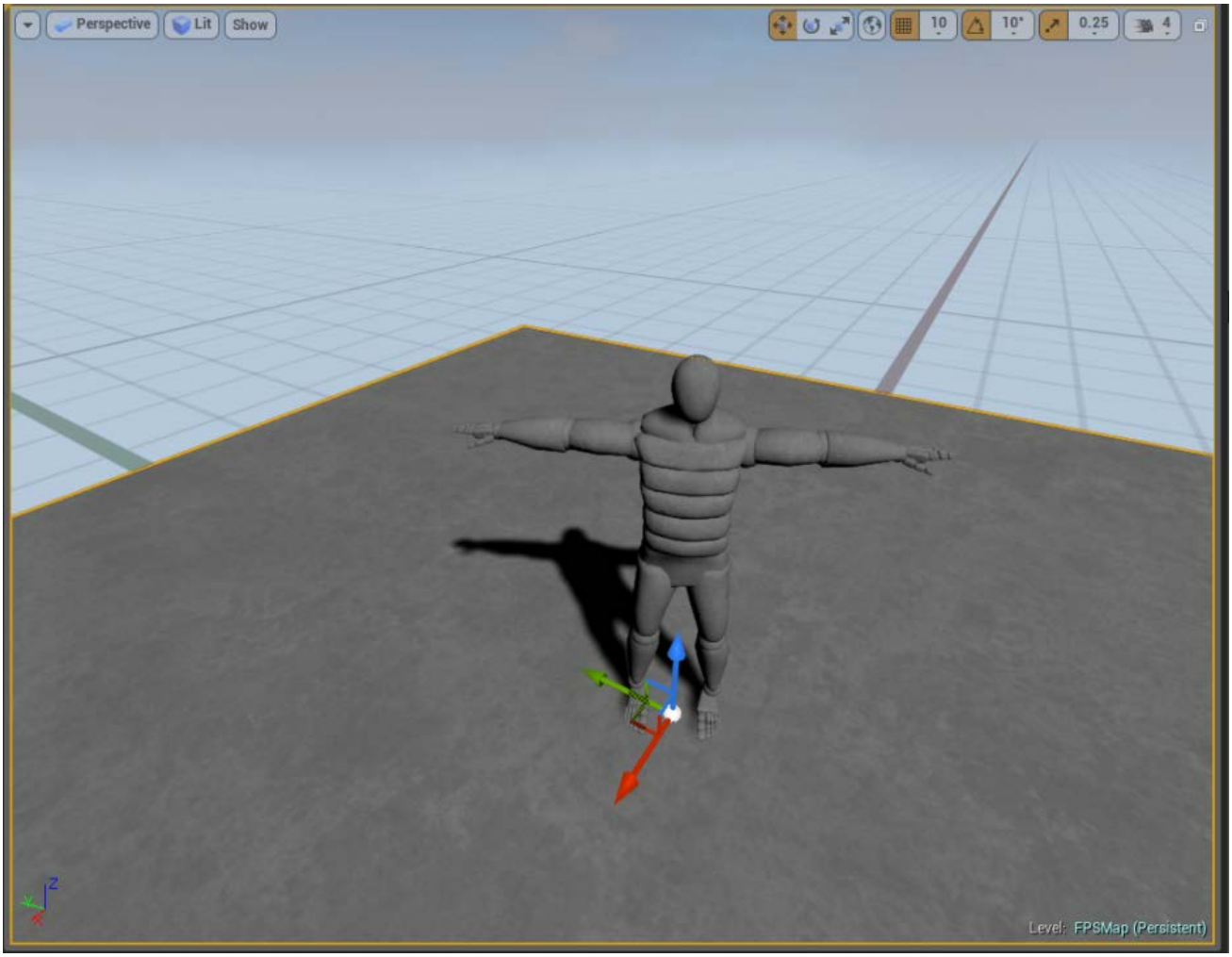

Рис. 4.62 - Меш в редакторі

5. Натисніть кнопку Зупинити, щоб вийти з режиму Play in Editor (PIE).

4.11 Урок №11: Зміна виду камери

В кінці попереднього кроку камера за замовчуванням розташована всередині шиї сітки. Давайте налаштуємо належну камеру, яку ми можемо використовувати для налаштування властивостей камери, таких як місцеположення та поле перегляду. Ми будемо робити це, додавши CameraComponent до нашого FPSCharacter. По-перше, давайте додамо властивість FPSCharacter для зберігання посилання на наш CameraComponent.

1. Перейдіть до FPSCharacter.h у Visual Studio та додайте наступне, щоб створити загальнодоступне властивість:

```
/** First person camera */
UPROPERTY(VisibleAnywhere, BlueprintReadOnly, Category=Camera)
UCameraComponent* FirstPersonCameraComponent;
```

Примітка: 4.18.2 Вам потрібно включити "Camera / CameraComponent.h".

Нам також буде потрібно додати конструктор до нашого файлу FPSCharacter.h.

```
// Constructor for AFPSCharacter
AFPSCharacter(const FObjectInitializer& ObjectInitializer);
```

Ми створимо фактичний компонент у конструкторі FPSCharacter.

2. Додайте наступний код в FPSCharacter.cpp, щоб створити CameraComponent та прикріпити його до CapsuleComponent.

```
AFPSCharacter::AFPSCharacter(const FObjectInitializer&
ObjectInitializer)
    : Super(ObjectInitializer)
{
    // Create a CameraComponent
```

```
      FirstPersonCameraComponent =
ObjectInitializer.CreateDefaultSubobject<UCameraComponent>(this,
TEXT("FirstPersonCamera"));
      FirstPersonCameraComponent->AttachParent = CapsuleComponent;
   }
```

Примітка: 4.11.2 CapsuleComponent недоступний, скористайтеся використанням GetCapsuleComponent ()

Примітка: 4.13 AttachParent тепер приватний. Використовуйте AttachTo ()

Нарешті, давайте налаштуємо позицію камери трохи вище розташування персонажа.

3. Додайте це до конструктора після створення компонента. Пізніше також можна налаштувати положення камери в BP_FPSCharacter Blueprint, але це дає хороше початкове місце для CameraComponent. Крім того, ви лише встановите місце CameraComponent, а не його обертання, тому що наші попередні функції повороту та перегляду керуватимуть орієнтацією камери.

```
   // Position the camera a bit above the eyes
   FirstPersonCameraComponent->RelativeLocation = FVector(0, 0, 50.0f
+ BaseEyeHeight);
   // Allow the pawn to control rotation.
   FirstPersonCameraComponent->bUsePawnControlRotation = true;
```

Оскільки ми додали нову властивість, нам потрібно скомпілювати в Visual Studio, а не в редакторі.

1. Закрийте Unreal Editor.

2. Скомпілювати.

3. Після закінчення збірки відкрийте Unreal Editor, а потім відкрийте FPSProject.

4. Натисніть кнопку “Відтворити” на панелі інструментів редактора рівня.

5. Тепер ваша камера повинна бути над головою персонажа, і якщо ви дивитесь вниз, ви зможете побачити верхню частину голови вашої персонажа.

6. Натисніть клавішу Escape, щоб вийти з режиму Play in Editor (PIE).

4.12 Урок №12: Додавання мешу для виду від першої особи

Поширеним підходом FPS є використання двох окремих сіток. Одним з них є нормальна сітка повного тіла, яка використовується при перегляді персонажа з третьої особи, але прихованої під час першої особи. Друга - це сітка "зброя та руки", яка прикріплена до камери, і вона є видимою лише для гравця, коли гравець перебуває з першої точки зору.

Для цього ми збережемо існуючий компонент з назвою Mesh як нашу третю особу і створіть новий SkeletalMeshComponent як нашу сітку першої особи.

1. Спочатку додати публічну змінну до FPSCharacter.h, щоб зберегти посилання на цю нову сітку:

```
/** Pawn mesh: 1st person view (arms; seen only by self) */
UPROPERTY(VisibleDefaultsOnly, Category=Mesh)
USkeletalMeshComponent* FirstPersonMesh;
```

2. Потім, у конструкторі для FPSCharacter.cpp, ми додамо код для створення та налаштування цієї сіткою після нашого коду, який налаштовує FirstPersonCameraComponent:

```
    // Create a mesh component that will be used when being viewed
from a '1st person' view (when controlling this pawn)
    FirstPersonMesh =
ObjectInitializer.CreateDefaultSubobject<USkeletalMeshComponent>(this,
TEXT("FirstPersonMesh"));
    FirstPersonMesh->SetOnlyOwnerSee(true);         // only the owning
player will see this mesh
    FirstPersonMesh->AttachParent = FirstPersonCameraComponent;
    FirstPersonMesh->bCastDynamicShadow = false;
    FirstPersonMesh->CastShadow = false;
```

Примітка: 4.13 AttachParent тепер приватний. Використовуйте функцію AttachTo (). (Api Reference)

Ми використовуємо SetOnlyOwnerЗнайти тут, щоб вказати, що ця сітка видно лише для "володільного" гравця, в цьому випадку PlayerController, який володіє цим символом. Ми також встановили сітку, яка буде приєднана до камери. Нарешті, ми відключимо деяку екологічну відтінку, оскільки побачити тіні цих прикріплених до камери зброї виглядатиме дивним і знищить ілюзію.

3. Нарешті, нам доведеться змінити налаштування Mesh, існуючої третьої особи SkeletalMeshComponent. Додайте наступні рядки до конструктора, щоб встановити його видимість, щоб вона була прихована від власника.

```
// everyone but the owner can see the regular body mesh
Mesh->SetOwnerNoSee(true);
```

Як і раніше, ми будемо встановлювати сітки в Blueprint, тому давайте скомпонуємо і запустимо редактор.

1. Закрийте Unreal Editor.

2. Скомпілювати.

3. Після закінчення збірки відкрийте Unreal Editor, а потім відкрийте FPSProject.

4. Завантажте цей файл zip і розпакуйте його, щоб отримати файл FBX з сіткою першої особи, в якому міститься SkeletalMesh з простими руками.

https://d26ilriwvtzlb.cloudfront.net/1/15/HeroFPP.zip

5. Перейдіть до папки “Гра” у браузері вмісту.

6. Клацніть правою кнопкою миші в браузері вмісту та виберіть пункт Імпорт до / Гра у меню, що з'явиться.

7. Перейдіть до того місця, де ви зберегли файл FBX, потім виберіть його та натисніть кнопку Відкрити.

8. Відкрийте розкривне меню Додатково та переконайтеся, що позначено Імпорт матеріали, а потім натисніть Імпорт.

Якщо ви отримаєте помилку про вирівнювання груп, ви можете ігнорувати це. Ця сітка все ще працюватиме, щоб проілюструвати налаштування сітки першого користувача, і буде працювати з анімаціями, налаштованими на більш пізньому етапі.

1. Натисніть кнопку Зберегти в браузері вмісту, щоб зберегти нову скелетну сітку та пов'язані з нею активи.

2. Перейдіть назад до папки Blueprints у браузері вмісту.

3. Відкрийте BP_FPSCharacter і перейдіть до режиму компонентів.

(ПРИМІТКА. Для того, щоб знайти FirstPersonMesh у списку компонентів, вам потрібно буде видалити стиль BP_FPSCharacter та відновити його. Після цього ви зможете продовжуйте крок 4.)

4. Знайдіть новий доданий нами FirstPersonMesh. Зверніть увагу, що вона є дитиною FirstPersonCameraComponent, тому вона завжди буде прикріплена до камери. Можливо, вам доведеться розкривати спадне меню FirstPersonCameraComponent.

Ви можете зіткнутися з проблемою, де нові FirstPersonMesh та FirstPersonCameraComponent не відображаються у вашому проекті. Якщо це станеться з вами, ви можете видалити проект BP_FPSCharacter і створити його знову, як описано вище, і з'являться нові властивості.

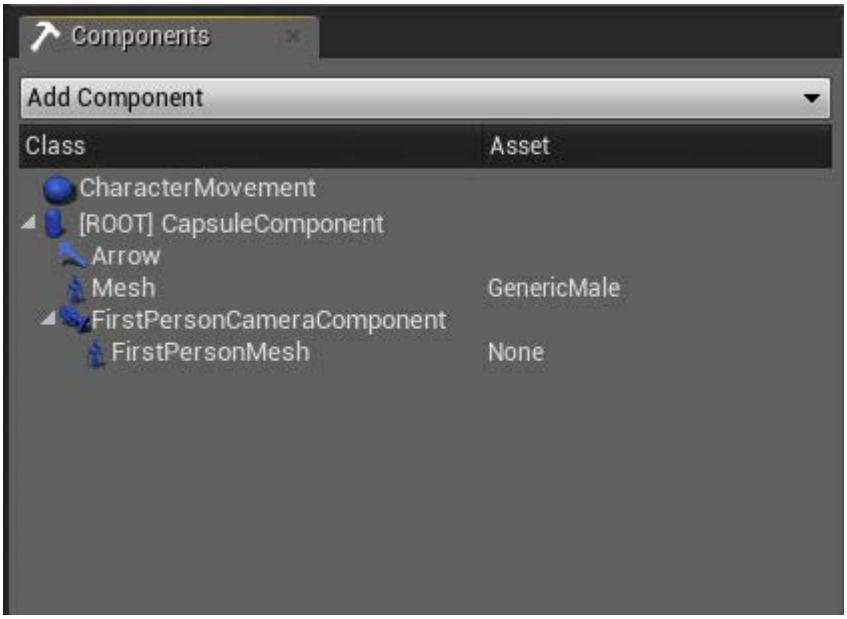

Рис. 4.63 - Додання компоненту

5. Клацніть компонент FirstPersonMesh на вкладці Компоненти.

6. На вкладці “Деталі” перейдіть до розділу “Мережа”. Натисніть спадне меню, яке говорить "Ніхто", а потім виберіть нещодавно завантажений актив "Скелетний сітка". Можливо, вам доведеться змінити розмір вкладки “Деталі”, щоб побачити це меню. Тепер, зброя має з'явитися на Viewport, хоча, можливо, доведеться зменшити масштаб, щоб побачити їх.

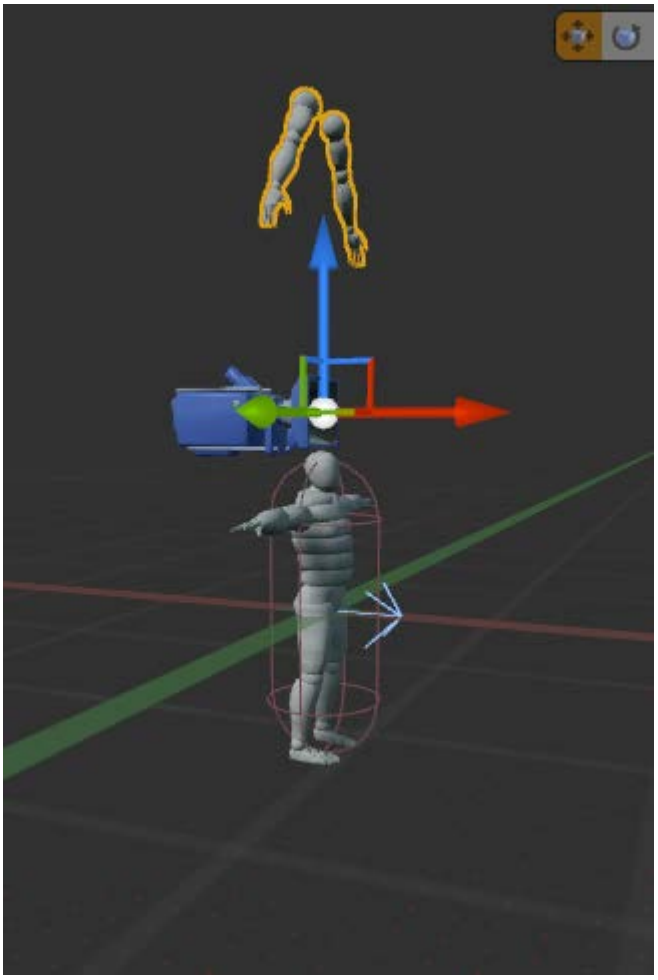

Рис. 4.64 - Налаштування мешу

7. Щоб настроїти відносне перетворення, щоб руки з'являлися на камері, встановіть для параметра "Місцеположення" значення {240,0,35} та "Обертання до" (-180, 50, -180). Ви відновите цю позицію після анімації зброї, але наразі ця позиція дозволяє вам побачити, що ви використовуєте цю сітку першої особи під час гри.

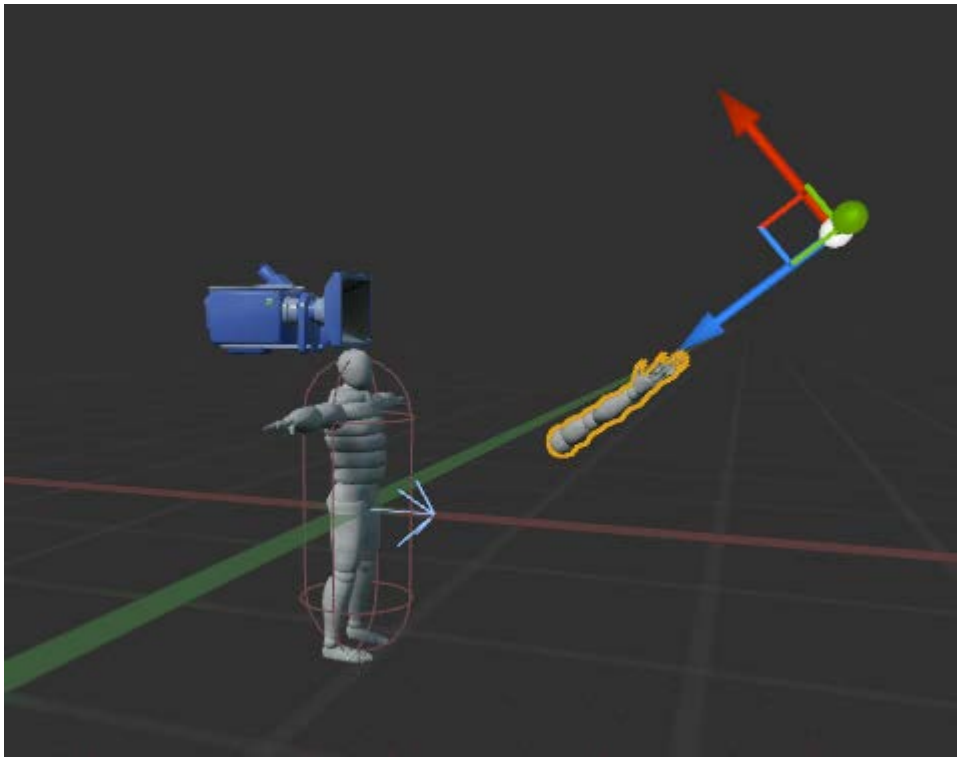

Рис. 4.65 - Налаштування мешу

8. Скомпілюйте та збережіть свій проект, а потім закрийте редактор креслень.

9. Натисніть кнопку “Відтворити” на панелі інструментів редактора рівнів.

10. На цьому етапі ви більше не зможете побачити третю особу SkeletalMesh, але ви зможете побачити беззмістовні руки першої особи SkeletalMesh.

Рис. 4.66 - Результат в редакторі

11. Натисніть Shift + F1, щоб відновити курсор миші, а потім натисніть кнопку Витягти на панелі інструментів. Ви більше не володієте персонажем, тому ви можете вільно переміщати камеру та побачити як третю особу, так і межі першої особи.

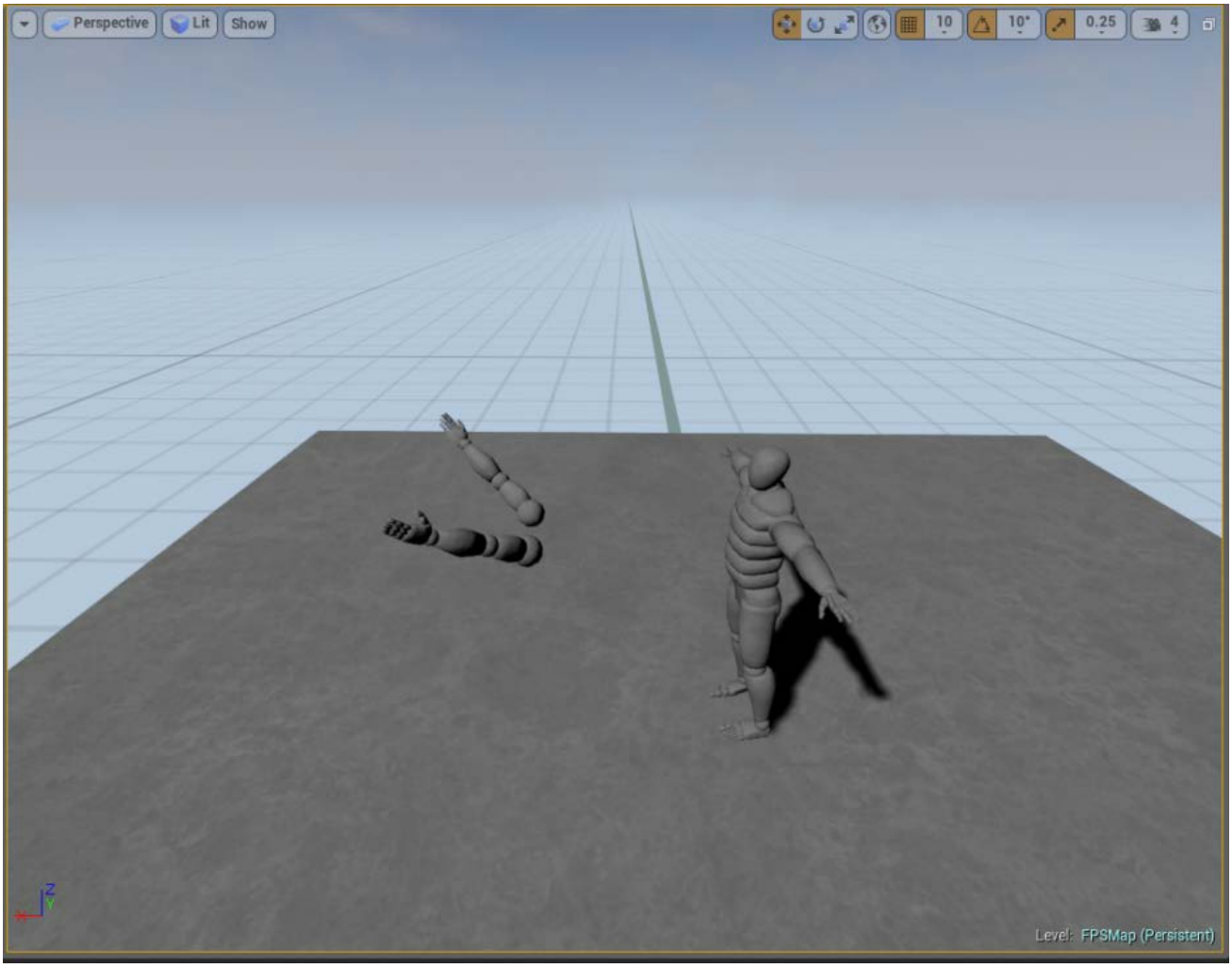

Рис. 4.67 - Результат в редакторі

12. Натисніть "Escape", щоб вийти з режиму Play in Editor (PIE).

4.13 Урок №13: Стрільба та пулі

Тепер, коли персонаж налаштований, давайте здійснимо просту зброю на снаряді - під час пожежі простої гранатоподібної снаряд буде стріляти з центру екрана і літати, поки вона не потрапить у світ. Поки ми відкриваємо редактор, давайте додамо вхід і створимо новий клас коду для нашого снаряда.

1. У меню “Редагування” клацніть Налаштування проекту.

2. У розділі "Двигун" в лівій частині вкладки Налаштування проекту натисніть на Вхід.

3. У розділі "Прив'язки" натисніть кнопку "+" біля пункту "Показів дій".

4. Клацніть на стрілці ліворуч від елементів дій, щоб розгорнути параметри мапування дій.

5. У текстовому полі введіть "Fire".

6. Розгорніть спадне меню і виберіть Ліва кнопка миші.

7. Закрийте меню Налаштування проекту.

Тепер давайте створимо клас снаряда.

1. Перейдіть до "Файл"> "Додати код до проекту".

2. Оберіть "Актор", потім натисніть "Далі".

3. Назвіть свій новий клас FPSProjectile, потім натисніть кнопку Створити.

4. Натисніть Так, щоб відкрити клас у Visual Studio для редагування.

5. Visual Studio запитає вас про перезавантаження проекту, оскільки майстер класів C ++ змінив його. Виберіть Перезавантажити.

По-перше, ми мусимо прийняти рішення щодо спрощеного фізичного представлення для використання при зіткненні та моделюванні. Для нашого випадку, давайте використовувати USphereComponent.

1. Додайте посилання на цей компонент у декларації класу FPSProjectile у FPSProjectile.h.

```
/** Sphere collision component */
UPROPERTY(VisibleDefaultsOnly, Category=Projectile)
USphereComponent* CollisionComp;
```

2. Додайте конструктор до FPSProjectile.h.

```
AFPSProjectile(const FObjectInitializer& ObjectInitializer);
```

3. Створіть компонент у конструкторі FPSProjectile у FPSProjectile.cpp. Ми зробимо це кореневим компонентом, оскільки симуляція буде керувати ним, і ми можемо долучити візуальні компоненти до нього пізніше в Blueprint.

```
    AFPSProjectile::AFPSProjectile(const FObjectInitializer&
ObjectInitializer)
        : Super(ObjectInitializer)
    {
        // Use a sphere as a simple collision representation
        CollisionComp =
ObjectInitializer.CreateDefaultSubobject<USphereComponent>(this,
TEXT("SphereComp"));
        CollisionComp->InitSphereRadius(15.0f);
```

```
        RootComponent = CollisionComp;
    }
```

UE4 поставляється з ProjectileMovementComponent, який може бути використаний для простоти руху простого балістичного стилю, так що давайте додамо, що до FPSProjectile.

1. По-перше, додайте загальну посилання в оголошенні класу FPSProjectile у FPSProjectile.h.

```
    /** Projectile movement component */
    UPROPERTY(VisibleAnywhere, BlueprintReadOnly, Category=Movement)
    UProjectileMovementComponent* ProjectileMovement;
```

2. Потім додайте наступні рядки до конструктора FPSProjectile у FPSProjectile.cpp для створення цього компонента:

```
    // Use a ProjectileMovementComponent to govern this projectile's
movement
    ProjectileMovement =
ObjectInitializer.CreateDefaultSubobject<UProjectileMovementComponent>
(this, TEXT("ProjectileComp"));
    ProjectileMovement->UpdatedComponent = CollisionComp;
    ProjectileMovement->InitialSpeed = 3000.f;
    ProjectileMovement->MaxSpeed = 3000.f;
    ProjectileMovement->bRotationFollowsVelocity = true;
    ProjectileMovement->bShouldBounce = true;
    ProjectileMovement->Bounciness  = 0.3f;
```

Ми також встановили кілька властивостей тут для впливу на симуляцію, найбільш важливим з яких є UpdateComponent.

Очікуючи вперед, ми також хочемо, щоб функція "запускала" наш снаряд, встановивши початкову швидкість.

1. Оголосіть загальну функцію в оголошенні класу FPSProjectile у FPSProjectile.h:

```
/** inits velocity of the projectile in the shoot direction */
void InitVelocity(const FVector& ShootDirection);
```

2. Впровадьте функцію в FPSProjectile.cpp, додаючи наступне визначення після конструктора.

```
void AFPSProjectile::InitVelocity(const FVector& ShootDirection)
{
    if (ProjectileMovement)
    {
        // set the projectile's velocity to the desired direction
        ProjectileMovement->Velocity = ShootDirection *
ProjectileMovement->InitialSpeed;
    }
}
```

Зверніть увагу, що швидкість нашого снаряда визначена в ProjectileMovementComponent, тому нам потрібно лише вказати напрямок запуску.

Нарешті, ми додамо код до FPSCharacter, так що при натисканні на Fire вводиться снаряд. Як і раніше, ми оголосимо загальнодоступну функцію, що називається OnFire, яку ми будемо пов'язувати з входом Fire, який ми визначили раніше.

1. У FPSCharacter.h додайте цю функцію до декларації класу:

```
//handles firing
UFUNCTION()
void OnFire();
```

2. В FPS Character.c ++ додайте наступне до SetupPlayerInputComponent для прив'язки дії Fire до нашої нової функції OnFire.

```
   InputComponent->BindAction("Fire", IE_Pressed, this,
&AFPSCharacter::OnFire);
```

Є два моменти, які слід враховувати для реалізації OnFire. Ми знаємо, що ми хочемо створити актора FPSProjectile, тому ми повинні визначити:

Де породити снаряд

Клас снаряда, таким чином, FPSCharacter (та його похідний проект) знатимуть, який снаряд виростити.

Щоб визначити розташування spawn, ми використовуватимемо офсетний вектор камери-космічного простору як редагований параметр, тому ми можемо встановити та налаштувати його в нашому BP_FPSCharacter Blueprint. Потім ми можемо обчислити початкове місце для снаряда на основі цих даних.

1. Додайте наступне до декларації класу FPSCharacter у FPSCharacter.h:

```
/** Gun muzzle's offset from the camera location */
UPROPERTY(EditAnywhere, BlueprintReadWrite, Category=Gameplay)
FVector MuzzleOffset;
```

Специфікатор EditAnywhere дозволяє змінити значення зміщення дурочки як у режимі за замовчуванням редактора Blueprint, так і на вкладці “Деталі” для будь-якого екземпляра символу. Специфікатор BlueprintReadWrite дозволяє вам як отримати, так і встановити значення зміщення дутра в межах проекту Blueprint.

Давайте також введемо клас снаряда як редагований параметр. Це дозволить нам пізніше визначити схему, отриману від FPSProjectile, як снаряд, який ми хочемо породжувати.

1. Додайте наступне до декларації класу FPSCharacter також у FPSCharacter.h:

```
    /** Projectile class to spawn */
    UPROPERTY(EditDefaultsOnly, Category=Projectile)
    TSubclassOf<class AFPSProjectile> ProjectileClass;
```

Тут ми використовуємо специфікацію EditDefaultsOnly, а це означає, що ви зможете встановити клас снаряда як типовий за замовчуванням на Blueprint, а не на кожному примірнику Blueprint.

Наша функція OnFire буде включати в себе кілька кроків.

Оскільки місце розташування нашого снаряда виводиться в космічному просторі камери, ми знаходимо перетворення камери перед розрахунком місця розташування спавнів.

Ми намагаємося створити снаряд.

Нарешті, ми даємо йому початкову швидкість, використовуючи визначену нами функцію.

1. Додайте наступну функцію до FPSCharacter.cpp:

```
    void AFPSCharacter::OnFire()
    {
        // try and fire a projectile
        if (ProjectileClass != NULL)
        {
            // Get the camera transform
            FVector CameraLoc;
            FRotator CameraRot;
            GetActorEyesViewPoint(CameraLoc, CameraRot);
            // MuzzleOffset is in camera space, so transform it to
world space before offsetting from the camera to find the final muzzle
position
            FVector const MuzzleLocation = CameraLoc +
FTransform(CameraRot).TransformVector(MuzzleOffset);
            FRotator MuzzleRotation = CameraRot;
            MuzzleRotation.Pitch += 10.0f;          // skew the aim
upwards a bit
            UWorld* const World = GetWorld();
            if (World)
            {
                FActorSpawnParameters SpawnParams;
                SpawnParams.Owner = this;
```

```
                SpawnParams.Instigator = Instigator;
                // spawn the projectile at the muzzle
                AFPSProjectile* const Projectile = World-
>SpawnActor<AFPSProjectile>(ProjectileClass, MuzzleLocation,
MuzzleRotation, SpawnParams);
                if (Projectile)
                {
                    // find launch direction
                    FVector const LaunchDir = MuzzleRotation.Vector();
                    Projectile->InitVelocity(LaunchDir);
                }
            }
        }
    }
```

2. Додайте #include у верхній частині FPSCharacter.cpp, оскільки ми використовуємо FPSProjectile з FPSCharacter.

```
#include "FPSProjectile.h"
```

3. Закрийте Unreal Editor.

4. Скомпонуйте.

5. Після закінчення збірки відкрийте Unreal Editor, а потім відкрийте FPSProject.

Тепер давайте завершимо наш снаряд, створивши Blueprint і додаючи сітку, щоб ми могли побачити ефект від стрілянини нашого снаряда.

6. Завантажте цей zip-файл і розпакуйте його, щоб отримати файл сфери FBX, який містить StaticMesh, який ви будете використовувати для вашого снаряда.

Сфера сітка

7. Перейдіть до папки “Гра” в браузері вмісту.

8. Клацніть правою кнопкою миші в браузері вмісту та виберіть меню Імпорт до / Гра у меню, що з'явиться.

9. Перейдіть до будь-якого місця, де ви зберегли файл FBX, потім виберіть його та натисніть кнопку Відкрити.

10. Відкрийте розкривне меню Додатково та переконайтеся, що позначено Імпорт матеріали, а потім натисніть Імпорт.

Якщо ви отримаєте помилку про вирівнювання груп, ви можете ігнорувати це. Ця сітка все ще працюватиме, щоб проілюструвати налаштування сітки першого користувача, і буде працювати з анімаціями, налаштованими на більш пізньому етапі.

11. Натисніть кнопку Зберегти в браузері вмісту, щоб зберегти новий статичний меш.

12. Перейдіть назад до папки Blueprints у браузері вмісту, потім клацніть правою кнопкою миші в браузері вмісту та створіть новий Blueprint.

13. Розкрийте спадне меню Custom Classes та знайдіть FPSProjectile, а потім виберіть його як батьківський клас для вашого проекту Blueprint.

14. Назвіть свій новий Blueprint BP_FPSProjectile, потім двічі клацніть по ньому, щоб відкрити його в редакторі Blueprint.

15. Щоб додати дитину StaticMeshComponent до кореневої області SphereComponent, натисніть на CollisionComp на вкладці “Компоненти”, а потім виберіть “Статична сітка” у спадному меню “Додати компонент”.

16. Назвіть цей новий компонент ProjectileMesh.

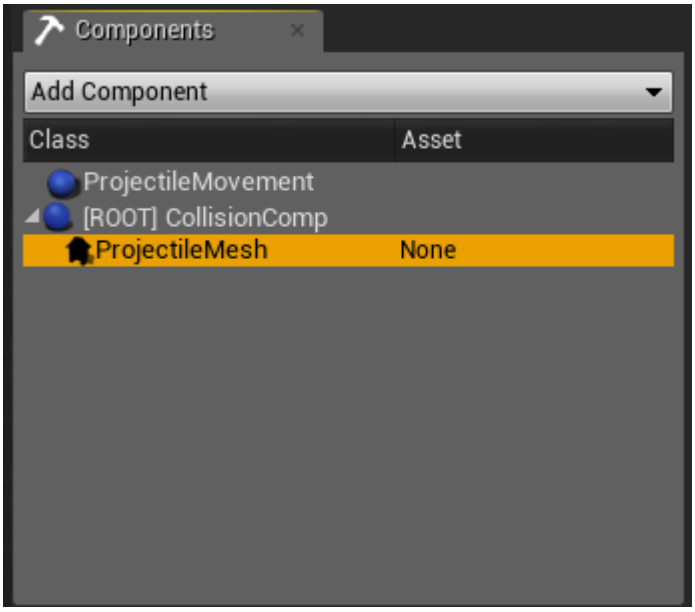

Рис. 4.68 - Додання мешу пострілу

17. Встановіть свій Статичний Mesh актив у вашу сферу StaticMesh за допомогою спадного меню під сіткою на панелі “Детальніше”.

Зверніть увагу, що якщо ви створюєте багатокористувацьку гру, то ви також повинні зніміть прапорець "Початкова швидкість у місцевому просторі" в компоненті "MovementComp", щоб цей снаряд правильно реплікував над сервером.

18. Встановіть масштаб на 0.09 в X, Y та Z.

Натискання на значок замка блокує всі три осей, щоб вони зберігали їх відносну шкалу.

19. Встановіть значення PresectibleMesh Collision Presets значення NoCollision, оскільки ми використовуємо SphereComponent для зіткнення, а не це Static Mesh.

20. Підготуйте та збережіть свій проект, а потім закрийте редактор креслень.

21. Тепер відкрийте BP_FPSCharacter для редагування та відкрийте режим за замовчуванням.

22. Знайдіть властивість класу Projectile і встановіть його на BP_FPSProjectile.

23. Встановіть властивість MuzzleOffset на {100, 0, 0}, щоб трохи породжувати снаряд перед камерою.

Рис. 4.69 - Параметри блупрінту

24. Скомпонуйте та збережіть свій проект, а потім закрийте редактор креслень.

25. Натисніть кнопку “Відтворити” на панелі інструментів редактора рівнів.

26. Клацніть лівою кнопкою миші, щоб звільнити снаряд

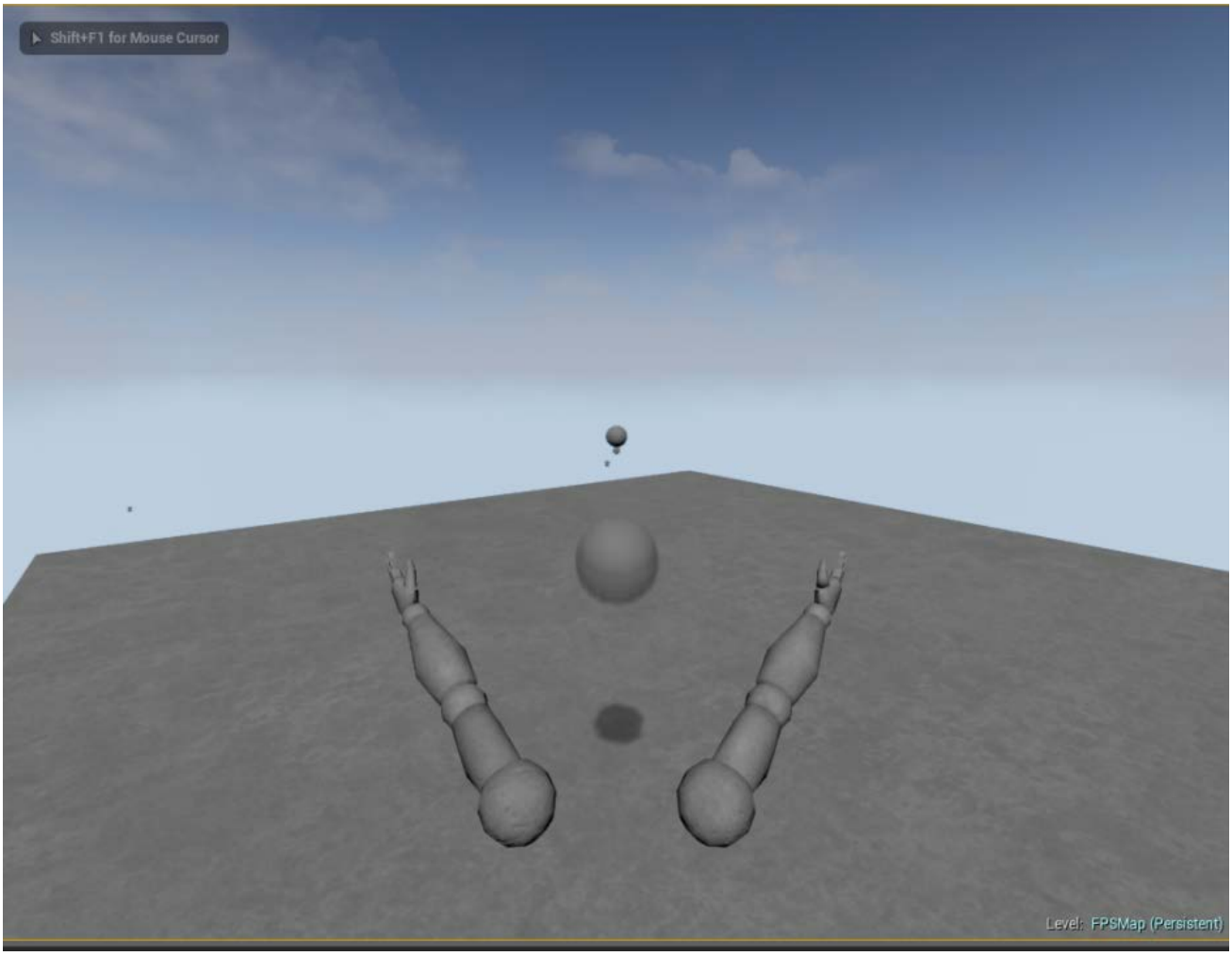

Рис. 4.70 - Результат в редакторі

27. Натисніть "Escape", щоб вийти з режиму Play in Editor (PIE).

4.14 Урок №14: Коллізія снаряду та час життя

Після додавання наших снарядів, є два цікаві речі, про які слід звернути увагу.

Боєприпаси живуть вічно, про що свідчить той факт, що вони залишаються назавжди у сцені Outliner.

Боєприпаси не стикаються ні з чим.

На щастя, всі учасники в UE4 можуть мати обмежений термін служби, що контролюється властивістю InitialLifeSpan.

1. Додайте наступні рядки до конструктора FPSProjectile, щоб встановити тривалість життя снаряда. Ви також можете змінити типовий початковий період життя в проекті BP_FPSProjectile.

```
// Die after 3 seconds by default
InitialLifeSpan = 3.0f;
```

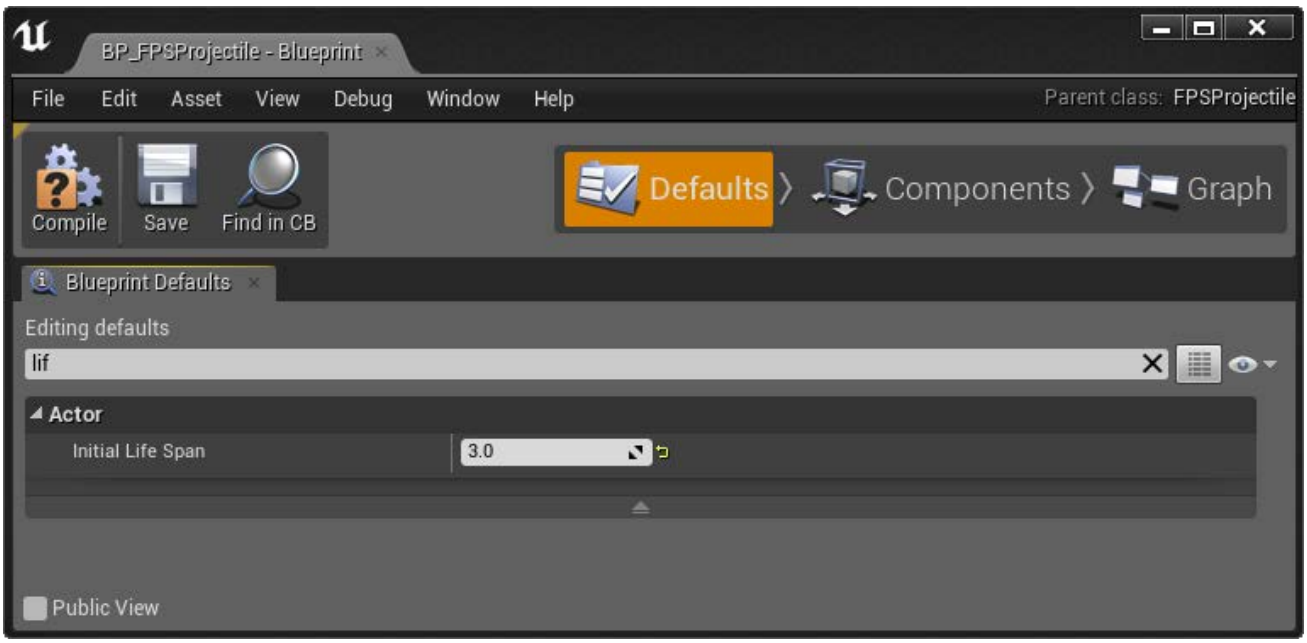

Рис. 4.71 - Редагування параметрів блупрінту

UE4 поставляється з декількома корисними каналами зіткнення, але також надає кілька настроюваних каналів, які можуть використовувати графічні проекти.

Давайте визначимо спеціальний канал для снарядів, тому всі зможуть явно вибрати, як взаємодіяти з снарядом в нашій грі.

1. Щоб настроїти канал, відкрийте Налаштування проекту та оберіть Collision.

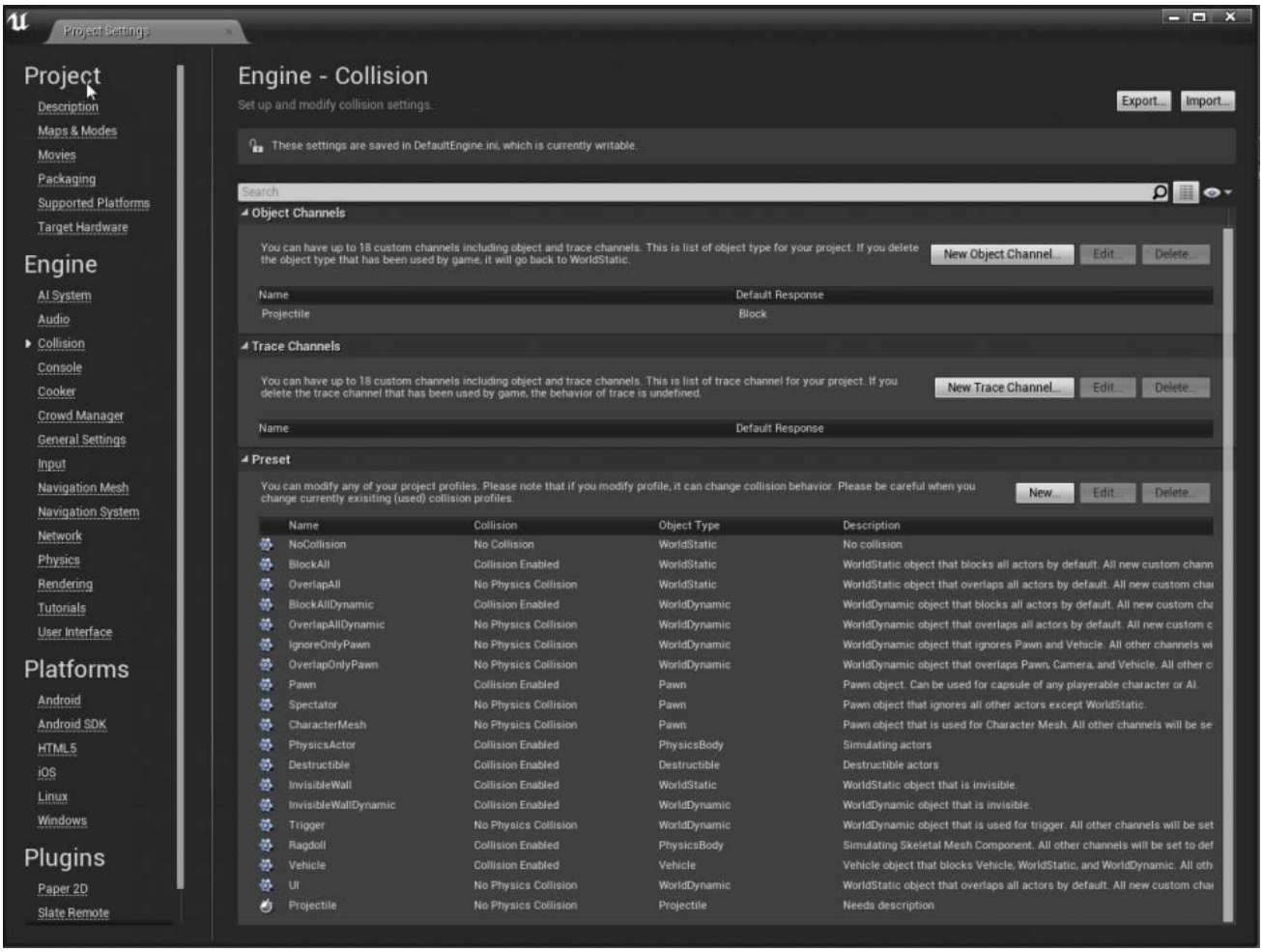

Рис. 4.72 - Налаштування колізії

2. Виберіть новий канал об'єктів ..., щоб створити новий канал зіткнень. Назвіть цей параметр і переконайтеся, що для параметра "Відповідь за замовчуванням" встановлено значення "Блок"

3. Тепер виберіть "Нове ..." у розділі "Пресет". Назвіть також цей Проджект. Установіть параметри для цієї стиснення зіткнень, як показано на зображенні нижче.

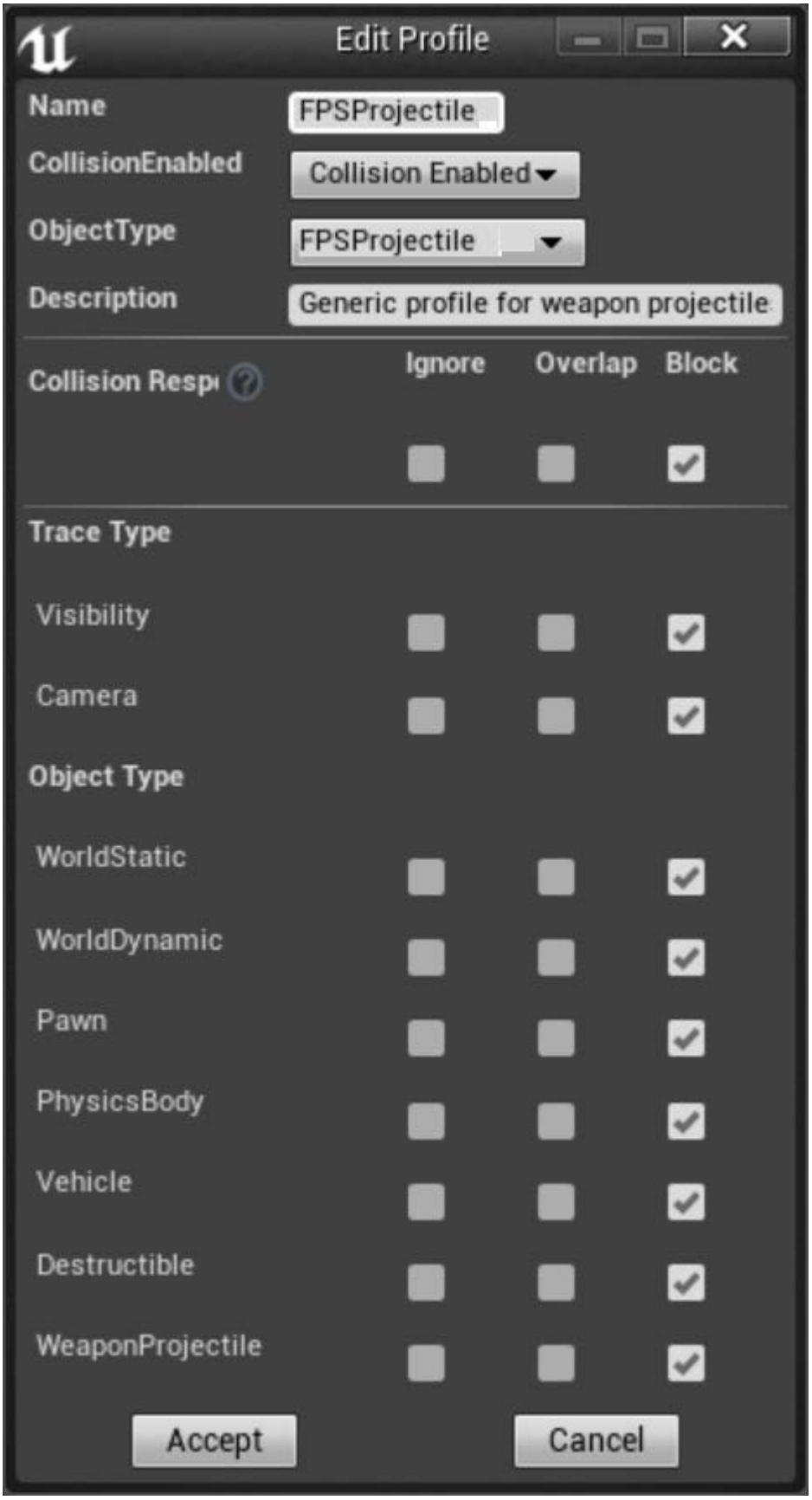

Рис. 4.73 - Налаштування профілю гри

Цей профіль означає, що снаряд буде заблоковано статичними акторами, пішаками, динамічними акторами, акторами, що імітують фізику, транспортні засоби та руйнуються учасники.

Тепер ми встановимо наш снаряд, щоб використовувати цей профіль.

1. У конструкторі FPSProjectile у FPSProjectile.cpp додати наступний рядок після створення CollisionComp.

```
CollisionComp->BodyInstance.SetCollisionProfileName("Projectile");
```

2. Закрийте Unreal Editor.

3. Скомпонуйте.

На наступному кроці ми визначимо, як снаряд взаємодіє з об'єктами, з якими стикається.

4.15 Урок №15: Взаємодія снаряду з об’єктами

Тепер, коли ми можемо виявити взаємодію зіткнення наших снарядів, ми можемо визначити, як відповісти на них. У наших налаштуваннях зіткнення ударів ми встановили, що наші взаємодії є блоками, тому додамо функцію FPSProjectile під назвою OnHit, щоб відповісти на ці події.

1. У визначенні класу FPSProjectile додайте наступну декларацію функції:

```
   /** called when projectile hits something */
   UFUNCTION()
   void OnHit(class AActor* OtherActor, class UPrimitiveComponent*
OtherComp, FVector NormalImpulse, const FHitResult& Hit);
```

Теперь ми впровадимо OnHit з функціональністю, щоб додати фізичний імпульс до будь-якого удару снаряда.

1. Додайте наступне до FPSProjectile.cpp:

```
   void AFPSProjectile::OnHit(AActor* OtherActor, UPrimitiveComponent*
OtherComp, FVector NormalImpulse, const FHitResult& Hit)
   {
       if ( OtherActor && (OtherActor != this) && OtherComp )
       {
           OtherComp->AddImpulseAtLocation(ProjectileMovement-
>Velocity * 100.0f, Hit.ImpactPoint);
       }
   }
```

Нарешті, нам потрібно зачепити цю функцію для делегата OnComponentBeginOverlap, що знаходиться на SphereComponent снаряда.

1. У конструкторі FPSProjectile додайте наступне після створення CollisionComp.

```
   CollisionComp->OnComponentHit.AddDynamic(this,
&AFPSProjectile::OnHit);
```

2. Скомпілювати.

3. Після закінчення збірки відкрийте Unreal Editor, а потім відкрийте FPSProject.

4. Виберіть поверх Floor StaticMesh SM_Template_Map_Floor.

5. Скопіюйте та вставте сітку підлоги, а потім скиньте копію до {0.2, 0.2, 3.0}.

6. Покладіть копію сіткою на {-230, 0, 160}.

7. Прокрутіть вниз до розділу Physics та перевірте Simulate Physics.

8. Збережіть свою карту.

9. Натисніть кнопку “Відтворити” на панелі інструментів редактора рівнів.

10. Клацніть лівою кнопкою миші, щоб звільнити снаряд і перемістити куб навколо вашого рівня. Ваші снаряди зараз завершені!

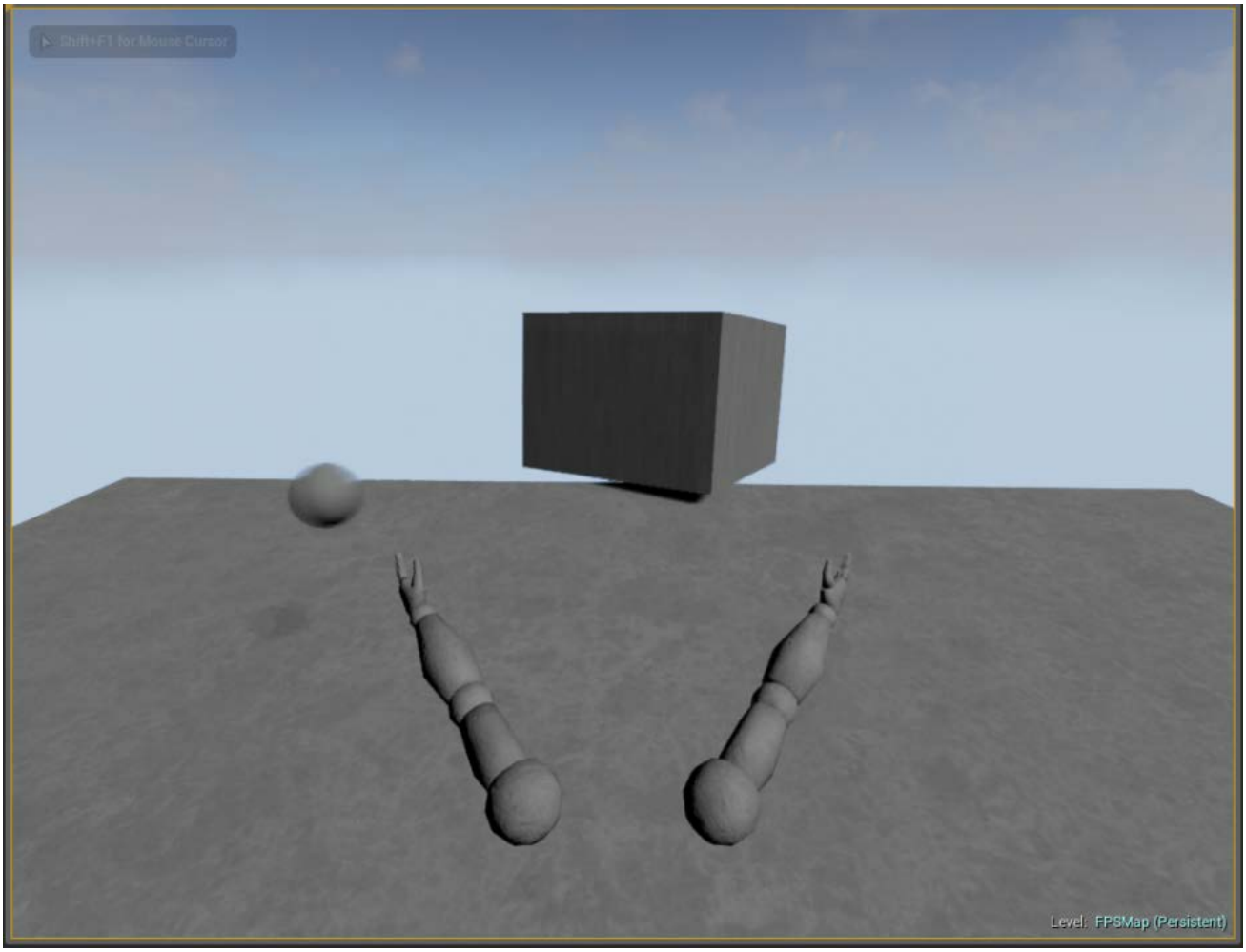

Рис. 4.74 - Результат в редакторі

4.16 Висновок

В даному розділі було створено програму курсу “Розробка комп’ютерних ігор”. Для цього було використано ігровий двигун Unreal Engine 4.

В програмі курсу розкриті такі теми, як:

- Встановлення Unreal Engine 4
- Створення проекту
- Навігація в інтерфейсі
- Імпортування ассетів
- Додавання мешів на рівень
- Матеріали

- Блупрінти
- Створення GameMode
- Створення персонажу
- Переміщення та стрибки
- Контроль камери
- Стрільба та взаємодія кулі з об’єктами

5 РОЗРОБЛЕННЯ СТАРТАП-ПРОЕКТУ “Курс створення комп’ютерних ігор”

5.1 Опис ідеї проекту

У даному розділі описано економічне обґрунтування реалізації стартап-проекту на тему “Курс створення комп’ютерних ігор”. В проекті пропонується розробка, яка дозволить створити нові кадри в сфері розробки комп’ютерних ігор на позиції дизайнерів (левел-дизайнер, технічний дизайнер, сторі-дизайнер) та програмістів (геймплей, фізика, штучний інтелект, нетворк, редактор, анімації).

Таблиця 5.1 - Опис ідеї стартап-проекту

Зміст ідеї	*Напрямки застосування*	*Вигоди для користувача*
Ідея полягає в тому, щоб створити програму для студентів старших курсів, що дозволить їм отримати специфічні навички для розробки комп’ютерних ігор	Для студентів вищих навчальних закладів	Можливість створення своєї гри або отримання роботи в ігровій компанії
	Як окремі платні курси для людей що зацікавлені	Можливість створення своєї гри або отримання роботи в ігровій компанії

Курс може бути використаний як в вищих навчальних закладах для студентів, так і як окрема підготовка для людей, що хочуть навчитися розробляти комп’ютерні ігри будь-якого жанру.

В таблиці 5.2 буде проаналізовано сильні, слабкі та нейтральні харатеристики ідеї курсу “Розробка комп’ютерних ігор”, що була наведена в таблиці 5.1. Це дозволить отримати інформацію про можливості для даного проекту по відношенню до конкурентів.

Таблиця 5.2 - Визначення сильних, слабких та нейтральних характеристик ідеї проекту

№	*Технікоекономічні*	*(Потенційні) товари/концепції*	*W*	*N*	*S*

п/п	*характеристики ідеї*	*конкурентів*				*(слабка сторона)*	*(нейтральна сторона)*	*(сильна сторона)*
		Мій проект	*Конкурент1*	*Конкурент2*	*Конкурент3*			
1.	Форма виконання	Серія уроків	Відео уроки	Текстові уроки	Серія уроків			+
2.	Собівартість	Середня	Середня	Низька	Середня		+	
3.	Ефективність	Висока	Середня	Низька	Середня			+
4.	Необхідна кількість часу	Велика	Середня	Дуже велика	Велика		+	

Визначений перелік слабких, сильних та нейтральних характеристик та властивостей ідеї потенційного товару є підґрунтям для формування його конкурентоспроможності. З таблиці видно, що запропоноване рішення є найбільш ефективним поміж конкурентів, а також тип подачі інформації є найкращим серед конкурентів, що доводить конкурентоспроможність даного проекту.

5.2 Технологічний аудит ідеї проекту

В межах даного підрозділу необхідно провести аудит технології, за допомогою якої можна реалізувати ідею проекту (технології створення товару).

Таблиця 5.3 - Технологічна здійсненність ідеї проекту

№ п/п	*Ідея проекту*	*Технології її реалізації*	*Наявність технології*	*Доступність технології*
1.	Створення курсу	Unreal Engine 4	Наявна	Умовно платна, доступна, широко використовується, підходить по технологіям

		CryEngine 5	Наявна	Умовно платна, доступна, середнє поширення, підходить по технологіям
		Unity	Наявна	Умовно платна, доступна, широко використовується, слабко підходить по технологіям
		Власний двигун	Потрібна розробка	Безкоштовна, доступна, нульове поширення, ідеально підходить по технологіям

Обрана технологія реалізації ідеї проекту – Unreal Engine 4, оскільки розробка власного двигуну майже не під силу одній людині (комерційні ігрові двигуни розробляються сотнями людей). Unity більше орієнтований на мобільний ринок, в той час як курс орієнтований на ринок персональних комп'ютерів та консолей. CryEngine 5 також являє собою гарну альтернативу, проте порівняно небагато людей знайомі з ним, що додасть проблем в пошуку відповідей на питання по двигуну в інтернеті.

5.3 Аналіз ринкових можливостей

Визначення ринкових можливостей, які можна використати під час ринкового впровадження проекту, та ринкових загроз, які можуть перешкодити реалізації проекту, дозволяє спланувати напрями розвитку проекту із урахуванням стану ринкового середовища, потреб потенційних клієнтів та пропозицій проектів-конкурентів.

Спочатку проводимо аналіз попиту: наявність попиту, обсяг, динаміка розвитку ринку.

Таблиця 5.4 - Попередня характеристика потенційного ринку стартап-проекту

№ п/п	*Показники стану ринку (найменування)*	*Характеристика*
1.	Кількість головних гравців, од	3
2.	Загальний обсяг продаж, грн/ум.од	20000 грн./ум.од
3.	Динаміка ринку (якісна оцінка)	Зростає
4.	Наявність обмежень для входу (вказати характер обмежень)	Немає
5.	Специфічні вимоги до стандартизації та сертифікації	Немає
6.	Середня норма рентабельності в галузі (або по ринку), %	R = (3000000 * 100) / (1000000 * 12) = 25%

Виходячи з таблиці 5.4, можна зробити висновки, що на ринку небагато головних гравців, що дозволяє сконцентрувати зусилля на порівнянні свого продукту з їх слабкостями. Позитивна динаміка ринку навчання говорить про актуальність розробки такого проекту и його необхідність на ринку.

Таблиця 5.5 - Характеристика потенційних клієнтів стартап-проекту

№ п/п	*Потреба, що формує ринок*	*Цільова аудиторія (цільові сегменти ринку)*	*Відмінності у поведінці різних потенційних цільових груп клієнтів*	*Вимоги споживачів до товару*
1.	Високий поріг входу в сферу розробки ігор	Будь-які люди, що зацікавленні в розробці гри	Цільова група, відмінностей між групами не	Рішення повинне бути зручним у

			має, всі можуть використовувати продукт	користуванні, надійним, ефективним

Судячи з таблиці 5.5, на ринку є потреба в наведеному продукті, оскільки високий поріг входу в сферу не дозволяє ефективно проводити самонавчання. Саме тому продукт може бути використаний для будь-якої групи населення, що зацікавлена в даній сфері.

Таблиця 5.6 - Фактори загроз

№ п/п	*Фактор*	*Зміст загрози*	*Можлива реакція компанії*
1.	Конкуренція	Вихід на ринок великої компанії	вихід з ринку; запропонувати великій компанії поглинути себе; передбачити додаткові переваги власного сервісу для того, щоб повідомити про них саме після виходу міжнародної компанії на ринок
2.	Зміна потреб користувачів	Користувачам необхідний курс з розширеним набором знань	Передбачити можливість додавання нових уроків до створеного курсу
3.	Зміна популярності	Зміна популярності вибраного двигуну	Передбачити можливість переходу курсу на інші двигуни
4.	Матеріальна частина	Подорожчання послуг	Орієнтація на якість послуг, виграш на масштабах
5.	Маловідомість	Малопоширенність	Реклама, знижки на курс

		відомостей про курс	

Виходячи з таблиці можна зробити висновок, що є багато ризиків зв'язаних с заснуванням компанії, серед найбільш ймовірних – малопоширенність відомостей про курс та потрібність додання додаткових матеріалів до курсу, проте було розроблено стратегії боротьби із цими ризиками, що повинні вивести .

Таблиця 5.7 - Фактори можливостей

№ п/п	*Фактор*	*Зміст можливості*	*Можлива реакція компанії*
1.	Зростання можливостей потенційних слухачів	Зростання фінансування у підприємств, які займаються предметами побуту	Запропонувати їм свої послуги
2.	Зниження довіри до конкурента	У конкурента було помічено негативні відгуки	При виході на ринок звертати увагу покупців на надійність нашого сервісу
3.	Залучення компаній	Люди будуть охочіше йти вчитися, якщо на курсах будуть представники ігрових компаній	Надавати ексклюзиву інформацію компаніям, що створюють комп'ютерні ігри

Є багато можливостей, котрі можна використати для поліпшення позицій компанії на ринку. Найбільш цікаво виглядає варіант залучення ігрових компаній до підготовки, що дозволить як покращити якість навчання, так і отримати спеціальні контракти з ігровими компаніями щодо працевлаштування.

Надалі проводиться аналіз пропозиції: визначаються загальні риси конкуренції на ринку.

Таблиця 5.8 - Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	*В чому проявляється дана характеристика*	*Вплив на діяльність підприємства (можливі дії*

		компанії, щоб бути конкурентоспроможною)
1. Вказати тип конкуренції: - досконала	Існує 3 фірми-конкуренти на ринку	Врахувати ціни конкурентних компаній на початкових етапах створення бізнесу, реклама (вказати на конкретні переваги перед конкурентами)
2. За рівнем конкурентної боротьби: - міжнародний	Всі компанії з інших країн	Додати можливість вибору курсу, щоб легше було у майбутньому вийти на міжнародний ринок
3. За галузевою ознакою: - внутрішньогалузева	Конкуренти мають ПЗ, який використовується лише всередині даної галузі	Створити основу ПЗ таким чином, щоб можна було легко його переробити для використання у інших галузях
4. Конкуренція за видами товарів: - товарно-видова	Види товарів є однаковими, а саме – навчання	Створити ПЗ, враховуючи недоліки конкурентів
5. За характером конкурентних переваг: - нецінова	Вдосконалення технології створення ПЗ, щоб собівартість була нижчою	Використання менш дорогих технологій для розробки, ніж використовують конкуренти
6. За інтенсивністю: - не марочна	Бренди відсутні	-

Ступеневий аналіз конкуренції на ринку наведений в таблиці 5.8 показує, що всі компанії-конкуренти з інших країн, що означає що український ринок фактично вільний для запропонованого продукту.

Методикою виділяються п'ять сил, які визначають рівень конкуренції, і, отже, привабливості ведення бізнесу в конкретній галузі.

Таблиця 5.9 – Аналіз конкуренції в галузі за М. Портером

Складові аналізу	*Прямі конкуренти в галузі*	*Потенційні конкуренти*	*Постачальники*	*Клієнти*	*Товари-замінники*
	Навести перелік прямих конкурентів	*Визначити бар'єри входження в ринок*	*Визначити фактори сили постачальників*	*Визначити фактори сили споживачів*	*Фактори загроз з боку замінників*
Висновки	Існує 3 конкуренти на ринку. Найбільш схожим за виконанням є конкурент 2, так як його рішення також представлене у вигляді набору уроків.	Так, можливості для входу на ринок є, оскільки в Україні подібних	Постачальники відсутні	Важливим для користувача є зручність у користуванні та ефективність курсу	Товари-замінники можуть зробити ефективнішу програму

Аналіз конкуренції в галузі за М. Портером для даного проекту дозволяє побачити можливі перешкоди з боку конкурентів. В даному випадку, замінники можуть зробити більш ефективну програму курсу, що дозволить їм випередити представлений проект. С огляду на це, потрібно буде постійно покращувати представлену пропозицію.

На основі аналізу конкуренції із урахуванням характеристик ідеї проекту, вимог споживачів до товару та факторів маркетингового середовища визначається та обґрунтовується перелік факторів конкурентоспроможності (табл. 4.10).

Таблиця 5.10 - Обґрунтування факторів конкурентоспроможності

№ п/п	*Фактор конкурентноспроможності*	*Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)*
1.	Цікавість інформації	Зацікавленість виростає в рази при вивченні інформації на практичній розробці гри
2.	Простота подачі	Інформація подається таким чином, щоб не було потрібно нічого запам’ятовувати, а потрібно було розуміти

Згідно з таблицею 5.10, можна побачити, що інформація подається таким чином, щоб не було потрібно нічого запам’ятовувати, а потрібно було розуміти, що дозволить вийти на інший рівень розуміння студентам представленого курсу.

За визначеними факторами конкурентоспроможності (табл. 4.10) проводиться аналіз сильних та слабких сторін стартап-проекту (табл. 4.11). Фінальним етапом ринкового аналізу можливостей впровадження проекту є складання SWOT-аналізу (матриці аналізу сильних (Strength) та слабких (Weak) сторін, загроз (Troubles) та можливостей (Opportunities) на основі виділених ринкових загроз та можливостей, та сильних і слабких сторін.

Таблиця 5.11 - Порівняльний аналіз сильних та слабких сторін проекту

№ п/п	*Фактор конкурентноспроможності*	*Бали 1-20*	*Рейтинг товарів-конкурентів у порівнянні з нашим підприємством*						
			3	*2*	*1*		*1*	*2*	*3*
.	Простота подачі	15							
	Ефективність курсу	20							

.									

Перелік ринкових загроз та ринкових можливостей складається на основі аналізу факторів загроз та факторів можливостей маркетингового середовища. Ринкові загрози та ринкові можливості є наслідками (прогнозованими результатами) впливу факторів, і, на відміну від них, ще не є реалізованими на ринку та мають певну ймовірність здійснення.

Таблиця 5.12 – SWOT-аналіз стартап-проекту

Сильні сторони: простота подачі, ефективність курсу, передовий двигун	Слабкі сторони: слабка рекламна компанія
Можливості: у конкурента 1 виявлена проблема із надійністю ПЗ, додаткове фінансування для розповсюдження даної технології	Загрози: конкуренція, зміна потреб користувачів, зміна популярності

На основі SWOT-аналізу розробляються альтернативи ринкової поведінки (перелік заходів) для виведення стартап-проекту на ринок та орієнтовний оптимальний час їх ринкової реалізації з огляду на потенційні проекти конкурентів, що можуть бути виведені на ринок. Визначені альтернативи аналізуються з точки зору строків та ймовірності отримання ресурсів.

Таблиця 5.13 – Альтернативи ринкового впровадження стартап-проекту

№ п/п	*Альтернатива (орієнтовний комплекс заходів) ринкової поведінки*	*Ймовірність отримання ресурсів*	*Строки реалізації*
1.	Створення на основі університету	80%	6 місяців
2.	Створення власної незалежної компанії	30%	12 місяців

З означених альтернатив обирається та, для якої: а) отримання ресурсів є більш простим та ймовірним; б) строки реалізації – більш стислими. Тому обираємо альтернативу 1.

5.4 Розробка ринкової стратегії проекту

Розроблення ринкової стратегії першим кроком передбачає визначення стратегії охоплення ринку: опис цільових груп потенційних споживачів.

Таблиця 5.14 - Вибір цільових груп потенційних споживачів

№ п/п	*Опис профілю цільової групи потенційних клієнтів*	*Готовність споживачів сприйняти продукт*	*Орієнтовни й попит в межах цільової групи (сегменту)*	*Інтенсивність конкуренції в сегменті*	*Простота входу у сегмент*
1.	Навчальні заклади	Можливість навчити студентів новим, цікавим та корисним дисциплінам	Великий	Існує 3 конкуренти, які надають схожі, але гірші рішення. Також усі вони знаходяться за межами країни.	Ефективність, простота подачі інформації, актуальність
2.	Зацікавлені люди (нові в галузі)	Можливість потрапити в бажану галузь	Великий		Ефективність, простота подачі інформації, актуальність
3.	Працівники ігрових компаній	Можливість підвищення кваліфікації	Середній		Ефективність, простота подачі

					інформації, актуальність

Виходячі з таблиці, можна зробити висновок, що найбільш доцільним буде вибрати такі цільові групи, як навчальні заклади, нові зацікавлені люди та працівникі игрових компаній.

За результатами аналізу потенційних груп споживачів (сегментів) автори ідеї обирають цільові групи, для яких вони пропонуватимуть свій товар, та визначають стратегію охоплення ринку.

Таблиця 5.15 – Визначення базової стратегії розвитку

№ п/п	*Обрана альтернатива розвитку проекту*	*Стратегія охоплення ринку*	*Ключові конкурентоспромо жні позиції відповідно до обраної альтернативи*	*Базова стратегія розвитку*
1.	Розвиток курсу на онлайн-основі	Ринкове позиціонування	Ефективність, простота у користуванні, простота доступу	Диференціація

Виходячі з таблиці, можна побачити, що було обрано ринкове позиціонування як стратегію охоплення ринку. Також, ключовими позиціями конкурентоспроможності, при виборі альтернативи в вигляді створення онлайн курсу, буде ефективність, простота у користуванні та простота доступу.

Таблиця 5.16 - Визначення базової стратегії конкурентної поведінки

№ п/п	*Чи є проект “першопрохідцем” на ринку?*	*Чи буде компанія шукати нових споживачів, або забирати існуючих*	*Чи буде компанія копіювати основні характеристики товару конкурента,*	*Стратегія конкурентної поведінки*

		у конкурентів?	*і які?*	
1.	На українському – Так На міжнародному - Ні	Так	Ні, не буде	Зайняття конкурентної ніші

Проект є першопрохідцем на українському ринку, це водночас і добре (через те, що нема конкурентів) і погано (через те, що нема прихильників даного підходу). В будь якому разі, планується не копіювати характеристики міжнародних конкурентів, а знайти свій шлях на цьому ринку.

На основі вимог споживачів з обраних сегментів до постачальника (стартап-компанії) та до продукту, а також в залежності від обраної базової стратегії розвитку та стратегії конкурентної поведінки розробляється стратегія позиціонування, що полягає у формуванні ринкової позиції (комплексу асоціацій), за яким споживачі мають ідентифікувати торгівельну марку/проект.

Таблиця 5.17 - Визначення стратегії позиціонування

№ п/п	*Вимоги до товару цільової аудиторії*	*Базова стратегія розвитку*	*Ключові конкурентоспроможні позиції власного стартап- проекту*	*Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)*
1.	Ефективність, простота у користуванні, простота доступу	Диференціація	Простота подачі дозволить та ефективність дозволить здобути прихільність нових студентів	Ефективність, простота подачі

В таблиці визначена стратегія позиціонування продукту, визначено вимоги, головною із яких є ефективність і простота курсу, що дозволить здобути прихільність нових студентів.

5.5 Розробка маркетингової програми

Першим кроком є формування маркетингової концепції товару, який отримає споживач. Для цього у табл. 5.18 потрібно підсумувати результати попереднього аналізу конкурентоспроможності товару.

Таблиця 5.18 - Визначення ключових переваг концепції потенційного товару

№ п/п	*Потреба*	*Вигода, яку пропонує товар*	*Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити)*
1.	Ефективніть	Курс розроблений так, щоб зацікавлення зростало на кожному уроці	Перевага у ефективності
2.	Простота доступу	Є варіанти як в онлайн форматі, так і в форматі курсу уроків	Користувачі мають зручний інтерфейс для взаємодії з навчальної програмою

Ключовими перевагами потенційного товару, згідно таблиці 5.18, є ефективність так простота доступу, що виділяються перед конкурентами.

Надалі розробляється трирівнева маркетингова модель товару: уточнюється ідея продукту та/або послуги, його фізичні складові, особливості процесу його надання (табл. 5.19).

Таблиця 5.19 - Опис трьох рівнів моделі товару

Рівні товару	*Сутність та складові*

<table>
<tr><td>I. Товар за задумом</td><td colspan="3">Курс розробки комп’ютерних ігор</td></tr>
<tr><td rowspan="4">II. Товар у реальному виконанні</td><td>Властивості/характеристики</td><td>М/Нм</td><td>Вр/Тх /Тл/Е/Ор</td></tr>
<tr><td>Простота викладання, ефективність, простота доступу</td><td>НЕ МАТЕРІАЛЬНА</td><td>ОРГАНІЗАЦІЙНА</td></tr>
<tr><td colspan="3">Якість: згідно до стандарту ISO 4444 буде проведено тестування</td></tr>
<tr><td colspan="3">Маркування відсутнє</td></tr>
<tr><td rowspan="2">III. Товар із підкріплюнням</td><td colspan="3">Моя компанія: “TheFirstLearning”</td></tr>
<tr><td colspan="3">2 пробних уроки</td></tr>
<tr><td colspan="4">Постійна підтримка для користувачів</td></tr>
<tr><td colspan="4">За рахунок чого потенційний товар буде захищено від копіювання: за рахунок унікальності подачі та інформації, що дозволить завоювати репутацію.</td></tr>
</table>

Три рівні моделі описують “Курс розробки комп’ютерних ігор”, компанія буде мати назву “TheFirstLearni"ng”, з додатковою підтримкою для користуваців і пробними уроками.

Після формування маркетингової моделі товару слід особливо відмітити – чим саме проект буде захищено від копіювання. Захист може бути організовано за рахунок захисту ідеї товару (захист інтелектуальної власності), або ноу-хау, чи комплексне поєднання властивостей і характеристик, закладене на другому та третьому рівнях товару.

Наступним кроком є визначення цінових меж, якими необхідно керуватись при встановленні ціни на потенційний товар (остаточне визначення ціни відбувається під час фінансово-економічного аналізу проекту), яке передбачає

аналіз ціни на товари-аналоги або товари субститути, а також аналіз рівня доходів цільової групи споживачів (табл. 5.20). Аналіз проводиться експертним методом.

Таблиця 5.20 - Визначення меж встановлення ціни

№ п/п	*Рівень цін на товари-замінники*	*Рівень цін на товари-аналоги*	*Рівень доходів цільової групи споживачів*	*Верхня та нижня межі встановлення ціни на товар/послугу*
1.	25000 грн	30000 грн	200000 грн	20000 грн

Наступним кроком є визначення оптимальної системи збуту, в межах якого приймається рішення (табл. 5.21).

Таблиця 5.21 - Формування системи збуту

№ п/п	*Специфіка закупівельної поведінки цільових клієнтів*	*Функції збуту, які має виконувати постачальник товару*	*Глибина каналу збуту*	*Оптимальна система збуту*
1.	Купують підписку та роблять щорічні внески для подовження	Продаж	0(напряму), 1(через одного посередника)	Власна та через посередників

Останньою складової маркетингової програми є розроблення концепції маркетингових комунікацій, що спирається на попередньо обрану основу для позиціонування, визначену специфіку поведінки клієнтів (табл. 5.22).

Таблиця 5.22 - Концепція маркетингових комунікацій

№ п/п	*Специфіка поведінки цільових клієнтів*	*Канали комунікацій, якими користуютьс я*	*Ключові позиції, обрані для позиціонування*	*Завдання рекламного повідомлення*	*Концепція рекламного звернення*

		цільові клієнти			
1.	Очні уроки	Заняття в класній кімнаті	Ефективність, простота подачі	Показати переваги сервісу, у тому числі і перед конкурентами	Демо-ролик із використання
2.	Використання за допомогою сайту	Інтернет	Простота доступу	Показати переваги сервісу, у тому числі і перед конкурентами	Демо-ролик із використання

Згідно таблиці, є декілька варінтів надання послуг, кожен із яких має свою специфіку, переваги так недоліки. Кожен користувач сам зможе вибрати, що йому більш потрібно, а також існує можливість змішаного курсу в використаннях обох методик.

5.6 Висновок

У даному розділі описано економічне обґрунтування реалізації стартап-проекту на тему “Курс створення комп’ютерних ігор”. В проекті пропонується розробка, яка дозволить створити нові кадри в сфері розробки комп’ютерних ігор на позиції дизайнерів (левел-дизайнер, технічний дизайнер, сторі-дизайнер) та програмістів (геймплей, фізика, штучний інтелект, нетворк, редактор, анімації).

Курс може бути використаний як в вищих навчальних закладах для студентів, так і як окрема підготовка для людей, що хочуть навчитися розробляти комп’ютерні ігри будь-якого жанру.

Запропоноване рішення є найбільш ефективним поміж конкурентів, а також тип подачі інформації є найкращим серед конкурентів, що доводить конкурентоспроможність даного проекту.

Обрана технологія реалізації ідеї проекту – Unreal Engine 4, оскільки розробка власного двигуну майже не під силу одній людині (комерційні ігрові двигуни розробляються сотнями людей). Unity більше орієнтований на мобільний ринок, в той час як курс орієнтований на ринок персональних комп'ютерів та консолей. CryEngine 5 також являє собою гарну альтернативу, проте порівняно небагато людей знайомі з ним, що додасть проблем в пошуку відповідей на питання по двигуну в інтернеті.

Позитивна динаміка ринку навчання говорить про актуальність розробки такого проекту и його необхідність на ринку. На ринку є потреба в наведеному продукті, оскільки високий поріг входу в сферу не дозволяє ефективно проводити самонавчання. Саме тому продукт може бути використаний для будь-якої групи населення, що зацікавлена в даній сфері.

Є багато можливостей, котрі можна використати для поліпшення позицій компанії на ринку. Найбільш цікаво виглядає варіант залучення ігрових компаній до підготовки, що дозволить як покращити якість навчання, так і отримати спеціальні контракти з ігровими компаніями щодо працевлаштування.

Аналіз конкуренції в галузі за М. Портером для даного проекту дозволяє побачити можливі перешкоди з боку конкурентів. В даному випадку, замінники можуть зробити більш ефективну програму курсу, що дозволить їм випередити представлений проект. С огляду на це, потрібно буде постійно покращувати представлену пропозицію.

Це все приводить к висновку, що створення подібного курсу є економічно обґрунтованим, а також, згідно наведених даних, є великий шанс того, що проект проявить себе дуже швидко та ефективно.

ВИСНОВКИ

В ході представленої роботи було досліджено місце комп’ютерних та мобільних ігор в сьогоднішньому світі. Було проведено пошукову роботу, що дала в результаті рейтинг найбільш відомих у світі курсів дизайну і розробки комп’ютерних ігор. В рейтингу було помічено дуже велику кількість серйозних університетів з великими грошовими вливаннями, наприклад, в США це Массачусетський технологічний інститут, Університет Південної Каліфорнії, Університет штату Юта, Мічиганський державний університет, Рочестерський технологічний інститут, Технологічний інститут DigiPen; у Великобританії це Імперський коледж Лондона, Університет Саутгемптона та Стаффордширський університет.

З огляду на кількість провідних шкіл для розробки відеоігор, не дивно, що рішення про продовження цієї дуже перспективної галузі навчання виявилося настільки популярним серед іноземних студентів. Вражаюча кількість добре профінансованих і високо поважних шкіл для розробки відеоігор у США надає дивовижний набір можливостей та кар'єрного потенціалу для іноземних студентів, які прагнуть зробити їхню прихильність до життя ігор у своїй професійній діяльності. Ті іноземні студенти, які вирішили вивчити розробку відеоігор у США, повинні вивчити деякі програми розробки відеоігор, що пропонуються в цих установах, щоб ознайомитись із конкретним підходом та зосередженням будь-якої майбутньої школи.

В ході роботи було розібрано існуючі ігрові двигуни – платформи для розробки ігор для комп’ютерів, телефонів, консолей і тд. Було виявлено найбільш поширені та найбільш потужні двигуни, якими стали Unity, Unreal Engine 4, Id Tech 4. Було проаналізовано кожен із них, його сильні та слабкі сторони. В результаті було отримано аналіз можливостей кожного з двигунів і його абстрактний рейтинг по відношенню до поставленої задачі гри. Було зроблено висновок, що Unreal Engine 4 найкраще підходить для розробки гри, з одним із найбільших недоліків у тому, що

в даному двигуні доволі високий поріг входу, що поставить деякі перешкоди перед новими програмістами та дизайнерами, проте ефективність двигуну перекриває цей недолік.

Було створено гру на вибраному двигуні Unreal Engine 4. За жанром гра є шутером від першого лиця з можливістю гри через мережу Інтернет, та з можливістю гри проти штучного інтелекту.

Також в роботі представлено курс “Розробка комп’ютерних ігор”, який складається з 15 уроків:

- Урок №1: Початок
- Урок №2: Інтерфейс, імпорт та додавання мешів на рівень
- Урок №3: Матеріали
- Урок №4: Блупрінти
- Урок №5: Створення GameMode
- Урок №6: Створення персонажу
- Урок №8: Контроль камери
- Урок №9: Стрибки
- Урок №10: Меш для персонажа
- Урок №11: Зміна виду камери
- Урок №12: Додавання мешу для виду від першої особи
- Урок №13: Стрільба та пулі
- Урок №14: Коллізія снаряду та час життя
- Урок №15: Взаємодія снаряду з об’єктами

Після проходження курсу, студент набуде базових навичок в роботі з Unreal Engine 4, зможе робити нескладні ігри та буде готовий до більш серйозного курсу “Просунута розробка комп’ютерних ігор”, в якому планується перейти до роботи з AI та нетворк технологіями.

ПЕРЕЛІК ПОСИЛАНЬ

1. University of Southern California [Електронний ресурс] / International Student. – Режим доступу до ресурсу: https://www.internationalstudent.com/school-search/141/usa/california/university-of-southern-california/.
2. University of Utah [Електронний ресурс] / International Student. – Режим доступу до ресурсу: https://www.internationalstudent.com/school-search/1025/usa/utah/university-of-utah/.
3. DigiPen Institute of Technology [Електронний ресурс] / International Student. – Режим доступу до ресурсу: https://www.internationalstudent.com/school-search/3663/usa/washington/digipen-institute-of-technology/.
4. Michigan State University [Електронний ресурс] / International Student. – Режим доступу до ресурсу: https://www.internationalstudent.com/school-search/481/usa/michigan/michigan-state-university.
5. Worcester Polytechnic Institute [Електронний ресурс] / International Student. – Режим доступу до ресурсу: https://www.internationalstudent.com/school-search/1384/usa/massachusetts/worcester-polytechnic-institute/.
6. Drexel University [Електронний ресурс] / International Student. – Режим доступу до ресурсу: https://www.internationalstudent.com/school-search/1564/usa/pennsylvania/drexel-university/.
7. Champlain College [Електронний ресурс] / International Student. – Режим доступу до ресурсу: https://www.internationalstudent.com/school-search/2237/usa/vermont/champlain-college/.
8. Rochester Institute of Technology [Електронний ресурс] / International Student. – Режим доступу до ресурсу: https://www.internationalstudent.com/school-search/2067/usa/new-york/rochester-institute-of-technology/.
9. Becker College [Електронний ресурс] / International Student. – Режим доступу до ресурсу: https://www.internationalstudent.com/school-search/1363/usa/massachusetts/becker-college/.

10. Rochester Institute of Technology [Електронний ресурс] / International Student. – Режим доступу до ресурсу: https://www.internationalstudent.com/school-search/2067/usa/new-york/rochester-institute-of-technology/.
11. Massachusetts Institute of Technology [Електронний ресурс] / International Student. – Режим доступу до ресурсу: https://www.internationalstudent.com/school-search/444/usa/massachusetts/massachusetts-institute-of-technology/.
12. University of Central Florida [Електронний ресурс] / International Student. – Режим доступу до ресурсу: https://www.internationalstudent.com/school-search/196/usa/florida/university-of-central-florida/.
13. Southern Methodist University [Електронний ресурс] / International Student. – Режим доступу до ресурсу: https://www.internationalstudent.com/school-search/999/usa/texas/southern-methodist-university/.
14. Carnegie Mellon University [Електронний ресурс] / International Student. – Режим доступу до ресурсу: https://www.internationalstudent.com/school-search/859/usa/pennsylvania/carnegie-mellon-university/.
15. Savannah College of Art and Design [Електронний ресурс] / International Student. – Режим доступу до ресурсу: https://www.internationalstudent.com/school-search/1843/usa/georgia/savannah-college-of-art-and-design/.
16. DigiPen Institute of Technology [Електронний ресурс] / International Student. – Режим доступу до ресурсу: https://www.internationalstudent.com/school-search/3663/usa/washington/digipen-institute-of-technology/.
17. University of California-Santa Cruz [Електронний ресурс] / International Student. – Режим доступу до ресурсу: https://www.internationalstudent.com/school-search/1189/usa/california/university-of-california-santa-cruz/.
18. Drexel University [Електронний ресурс] / International Student. – Режим доступу до ресурсу: https://www.internationalstudent.com/school-search/1564/usa/pennsylvania/drexel-university/.

19. Top Schools for Video Game Development [Електронний ресурс] / International Student. – Режим доступу до ресурсу: https://www.internationalstudent.com/study-video-game-development/top-schools-for-video-game-development/.
20. GAME PROGRAMMING DEGREE DRIVE THE FUTURE OF GAMES WITH VIDEO GAME PROGRAMMING DEGREE [Електронний ресурс] / University of Advancing Technology. – Режим доступу до ресурсу: http://www.uat.edu/game-programming-degree.
21. Computer Games Design and Programming BSc [Електронний ресурс] / Staffordshire University. – Режим доступу до ресурсу: http://www.staffs.ac.uk/course/SSTK-11161.jsp.
22. Game Programming [Електронний ресурс] / Brock University. – Режим доступу до ресурсу: https://brocku.ca/programs/undergraduate/game-programming/ .
23. Top Schools for Video Game Development [Електронний ресурс] / International Student. – Режим доступу до ресурсу: https://www.internationalstudent.com/study-video-game-development/top-schools-for-video-game-development/.
24. Game Programming [Електронний ресурс] / Manchester Metropolitan University. – Режим доступу до ресурсу: http://www2.mmu.ac.uk/study/undergraduate/courses/2016/13059/.
25. Games Design BA [Електронний ресурс] / Brunel University of London. – Режим доступу до ресурсу: http://www.brunel.ac.uk/courses/undergraduate/games-design-ba .
26. BA (Hons) Games Design and Art (3 years) [Електронний ресурс] / University of Southampton. – Режим доступу до ресурсу: http://www.southampton.ac.uk/wsa/undergraduate/courses/ba_games_design.page.
27. Computing - Games, Vision and Interaction (MEng) [Електронний ресурс] / Imperial College London. – Режим доступу до ресурсу: http://www.imperial.ac.uk/computing/prospective-students/courses/ug/beng-meng-computing/meng-comp-gvi/.

28. Unity (рушій гри) [Електронний ресурс] / Wikipedia. – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Unity_(%D1%80%D1%83%D1%88%D1%96%D0%B9_%D0%B3%D1%80%D0%B8).
29. Unreal Engine [Електронний ресурс] / Wikipedia. – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Unreal_Engine.
30. id Tech 4 [Електронний ресурс] / Wikipedia. – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Id_Tech_4.
31. Unreal Engine [Електронний ресурс] / Wikipedia. – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Unreal_Engine.